Universidad Politécnica de Madrid

Escuela Técnica Superior
de Ingenieros de Telecomunicación

Optimization of Protection Techniques Based on FEC Codes for the Transmission of Multimedia Packetized Streams

Tesis Doctoral

Filippo Casu
Ingeniero de Telecomunicación
2017

Departamento de Señales,
Sistemas y Radiocomunicaciones

Escuela Técnica Superior
de Ingenieros de Telecomunicación

Optimization of Protection Techniques Based on FEC Codes for the Transmission of Multimedia Packetized Streams

Tesis Doctoral

Autor:

Filippo Casu
Ingeniero de Telecomunicación
Universidad Politécnica de Madrid

Director:

Julián Cabrera Quesada
Doctor Ingeniero de Telecomunicación
Profesor del Departamento de Señales,
Sistemas y Radiocomunicaciones
Universidad Politécnica de Madrid

TESIS DOCTORAL

Optimization of Protection Techniques Based on FEC Codes for the Transmission of Multimedia Packetized Streams

Autor: Filippo Casu
Director: Julián Cabrera Quesada

Tribunal nombrado por el Sr. Rector Magnífico de la Universidad Politécnica de Madrid, el día ..... de ..... de 2017.

Presidente: D. .....
Vocal: D. .....
Vocal: D. .....
Vocal: D. .....
Secretario: D. .....

Realizado el acto de defensa y lectura de la Tesis el día ..... de ..... de 2017 en .....

Calificación: .....

EL PRESIDENTE LOS VOCALES

EL SECRETARIO

Resumen

Esta tesis presenta dos modelos novedosos de arquitecturas basadas en esquemas FEC con el fin de proteger flujos de paquetes con contenido multimedial, para comunicaciones en tiempo real y en canales donde las pérdidas se producen en ráfagas. El objetivo de estos diseños ha sido maximizar la eficiencia de los códigos FEC considerados. Por un lado, el primer modelo busca alcanzar un menor coste computacional para los códigos de Reed-Solomon, ya que su conocida capacidad de recuperación para todo tipo de canales necesita un coste computacional elevado. Por otro lado, en el caso de los códigos LDPC, se ha perseguido aumentar la capacidad de recuperación de estos códigos operando en canales con errores en ráfagas, teniendo en cuenta que los códigos LDPC no están directamente diseñados para este tipo de entorno.

El modelo aplicado a los códigos de Reed-Solomon se denomina *inter-packet symbol approach*. Este esquema consiste en una estructura alternativa que asocia los bits de los símbolos del código en distintos paquetes. Esta característica permite aprovechar de forma mejor la capacidad de recuperación de los códigos de Reed-Solomon frente a pérdidas de paquetes en ráfagas. Las prestaciones de este esquema han sido estudiadas en términos de tiempo de codificación/decodificación versus capacidad de recuperación y han sido comparados con otros esquemas propuestos en literatura. El análisis teórico ha demostrado que el enfoque propuesto permite la utilización de Campos de Galois de menor dimensión con respecto a otras soluciones. Esto se traduce en una disminución del tiempo de codificación/decodificación requerido, mientras que mantiene una capacidad de recuperación comparable.

Aunque la utilización de los códigos LDPC está típicamente orientada hacía canales con errores uniformemente distribuidos (canales sin memoria) y para bloques de información largos, esta tesis surgiere el uso de este tipo de códigos FEC a nivel de aplicación, para canales con pérdidas en ráfagas y para entornos de comunicación de tiempo real, es decir, con una latencia de transmisión muy baja. Para satisfacer estas limitaciones, la configuración apropiada de los parámetros de un código LDPC ha sido determinada usando bloques de información pequeños y adaptando el código FEC de modo que sea capaz de recuperar paquetes perdidos en canales con errores en ráfagas. Para ello, primeramente se ha diseñado un algoritmo que realiza una estimación de las capacidades de recuperación del código LDPC para un canal con pérdidas en ráfagas. Una vez caracterizado el código, se ha diseñado un segundo algoritmo que optimiza la estructura del código en términos de capacidad de recuperación para las características especificas

del canal con memoria, generado una versión modificada del código LDPC, adaptada al canal con perdidas en ráfagas.

Finalmente, los dos esquemas FEC propuestos, han sido evaluado experimentalmente en entornos de simulación usando canales con errores en ráfagas y se han comparado con otras soluciones y esquemas ya existentes.

Abstract

This thesis presents two enhanced FEC-based schemes to protect real-time packetized multimedia streams in bursty channels. The objective of these novel architectures has been the optimization of existing FEC codes, that is, Reed-Solomon codes and LDPC codes. On the one hand, the optimization is focused on the achievement of a lower computational cost for Reed-Solomon codes, since their well known robust recovery capability against any type of losses needs a high complexity. On the other hand, in the case of LDPC codes, the optimization is addressed to increase the recovery capabilities for a bursty channel, since they are not specifically designed for the scenario considered in this thesis.

The scheme based on Reed-Solomon codes is called inter-packet symbol approach, and it consists in an alternative bit structure that allocates each symbol of a Reed-Solomon code in several media packets. This characteristic permits to exploit better the recovery capability of Reed-Solomon codes against bursty packet losses. The performance of this scheme has been studied in terms of encoding/decoding time versus recovery capability, and compared with other proposed schemes in the literature. The theoretical analysis has shown that the proposed approach allows the use of a lower size of the Galois Fields compared to other solutions. This lower size results in a decrease of the required encoding/decoding time while keeping a comparable recovery capability.

Although the use of LDPC codes is typically addressed for channels where losses are uniformly distributed (memoryless channels) and for large information blocks, this thesis suggests the use of this type of FEC codes at the application layer, in bursty channels and for real-time scenario, where low transmission latency is requested. To fulfill these constraints, the appropriate configuration parameters of an LDPC scheme have been determined using small blocks of information and adapting the FEC code to be capable of recovering packet losses in bursty environments. This purpose is achieved in two steps. The first step is performed by an algorithm that estimates the recovery capability if a given LDPC code in a burst packet loss network. The second step is the optimization of the code: an algorithm optimizes the code structure in terms of recovery capability against the specific behavior of the channel with memory, generating a burst oriented version of the considered LDPC code.

Finally, for both proposed FEC schemes, experimental results have been carried out in a simulated transmission channel to assess the performances of the schemes and compared to several other schemes.

Agradecimientos

Primero quiero dar las gracias a mi tutor, Julián, que me ha soportado (e sopportato) todos estos años en el GTI, desde el Proyecto Fin de Carrera hasta terminar la tesis. Además, después de marcharme, le ha tocado seguir con esta tarea: grandes los momentos pasados charlando por Skype. La mayor parte de este trabajo se la debo a él y a toda la paciencia que ha tenido para corregir mi español y mi inglés. Que sepa que cuando necesite que alguien le corrija algo en italiano, aquí estaré.

Luego quiero darles las gracias a todos los profesores del GTI, por todo el aporte profesional y no profesional que me han dado estos años. En particular quiero mencionar a Narciso, por todo su ayuda con los artículos y todas las gestiones de las cuestiones burocráticas (como estudiante extranjero y como estudiante español) y también por tenerme siempre en cuenta a la hora de pedir el Pesquera antes de Navidad. A Fernando, por su infinita disponibilidad, desde aquel día en el tribunal para la defensa del PFC: gracias por no haberme tratado mal delante de mi madre. A Luis, por todas aquellas veces en el teatro escuchando buena música. A F., por haberme regalado "Spain" de Tomatito y Michel Camilo nada más entrar en el GTI y todas las cosas que me ha enseñado de la España cañí.

Ahora pasamos a agradecer a los miembros más "jóvenes" del GTI. A César, por haber estado siempre, desde el principio principio, hasta el final final (le toca corregir estos agradecimientos), por todo el español que he aprendido con él (llevar, traer), por su apoyo dentro del GTI y fuera, cuando le tocaba administrar el bote en los festivales. A Massi y Gianlu, mis compañeros italianos que me han abandonado (nunca os lo voy a perdonar), pero que han sido mi pequeña familia italiana en el grupo. Además, Gianlu fue quien me llevó al GTI (todo es por su culpa). A Jesús y Raúl, que de compañeros de curro se convirtieron en amigos y luego en algo más que una familia en Encomienda9, una referencia social, cultural, culinaria para todo el bonito barrio de Lavapiés. A Pablo, por preferirme cuando estaba más gordito y venir a Roma a ver a los Radiohead conmigo. A Dani, por la integral de los cuartetos de Shostakovich, para que yo entendiera que no se compone de espaldas al pueblo. A Carlos Roberto, el maestro, por haberme llevado a esta locura de correr maratones. Y a toda esta multitud de gente que se ha pasado por el GTI, como Maykel y Virgini a (por aguantarme como vecino), Sasho, por su sonrisa, Carlos Cuevas, por la historia de pintar pasos de bailes en el pavimento, Mamma Esther (hicimos el PFC en la misma época, y ahora...), Richard, Guillermo, Jon, los nuevos (que ya no son tan nuevos) Rafa, Tomás, Ana, Susana, Sergio, etc, etc, etc. Si se me olvida

alguien es culpa de las correcciones de César.

Quiero dar la gracias a todos mis amigos "madrileños", que han pasado por mi vida en estos años y que han contribuido de alguna forma a que este trabajo se llevara a cabo. Hay mogollón de italianos, franceses, ingleses, chipriotas, gallegos, valencianos, murcianos, gente de todos los colores, razas y especias (¡especies!). En particular quiero darle las gracias a Jorge, por haber sido una presencia fija en mi vida en Madrid, desde el primer año.

Ahora los sardos! A mi familia primero, por todo el apoyo durante estos años, y en particular a mi hermana Simonetta, a mi tío Eliseo y a mi madre, que me ha empujado a terminar la tesis, aunque, estando aquí en la isla bonita, no siempre ha sido fácil. A mis amigos de siempre, Michi y Fabio. Y finalmente a Anna, que aguantar a Filippo no es nada moco de pavo, y aguantar a Filippo_escribiendo_una_tesis aún menos.

Muchas gracias a todos por (o para?) todo.

List of Abbreviations

FEC	Forward Error Correction
LDPC	Low-Density Parity-Check
RS	Reed-Solomon
ARQ	Automatic Repeat reQuest
IP	Internet Protocol
VOIP	Voice Over IP
AL	application layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WWW	World Wide Web
RTP	Real-time Transport Protocol
HARQ	Hybrid Automatic Repeat reQuest
XOR	Exclusive OR
GF	Galois Field
1-D	1-Dimensional
2-D	2-Dimensional
LT	Luby Transform
MDS	Maximum Distance Separable
RFC	Request For Comments
LDGM	Low-Dense Generator-Matrix
PEC	Packet Erasure Channels

CRC	Cyclic Redundant Check
SN	Sequence Number
IETF	Internet Engineering Task Force
PER	Packet Error Rate
CRM	Column-based Recovery Measurement
LDBOGM	Low-Density Burst-Oriented Generator Matrix
GRM	Global Recovery Measurement
OSI	Open Systems Interconnection
DSL	Digital Subscriber Lines
GoP	Group of Pictures
AVC	Advanced Video Coding
PSNR	Peak Signal-to-Noise Ratio
HEVC	High Efficiency Video Coding
CBR	Constant Bit Rate

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Nowadays, multimedia communications are moving toward IP (Internet Protocol) networks, since they are more efficient than circuit-switching channels. The most popular real-time applications that deliver services using IP networks are represented by voice over IP (VOIP), television over IP, and video conference.

Nevertheless, IP networks have some weak points (e.g., unreliable packets delivery and delay control problems) that might damage the media content. For this reason, research is centring its efforts on developing efficient and robust systems against delays and channel errors.

Specifically, errors and losses can occur at different levels: data link layer and higher layers. In this sense, those at the data link layer are originated by low values of signal-to-noise ratio (bits arrive at the receiver either correctly or erroneously). In this layer, error detection and error correction are possible with the use of forward error correction (FEC) codes. On the other hand, at higher layers, losses and errors also occur due to sporadic impulse noise or congestion of the network routers, leading to bursts of packet losses (erasure channels). In this case, the use of application layer FEC (AL-FEC) or automatic repeat request (ARQ) schemes can prevent these errors.

IP is usually employed with the transport level protocols, namely transmission control protocol (TCP) and user datagram protocol (UDP). TCP provides reliable and ordered packet delivery and is employed by many popular Internet applications, such as World Wide Web (WWW), E-Mail, etc. UDP is commonly used in time-sensitive applications, since it allows lower delays, thanks to a simpler transmission model. Nevertheless, UDP provides unreliable communication: neither the reception of packets nor the order of the received packets are guaranteed. Hence, it is necessary to include additional mechanisms to give reliability to the communication.

In this sense, a typical solution is partly represented by the real-time transport protocol (RTP), an application layer protocol that is generally used over UDP. RTP defines a packet format for delivering data with real-time characteristics, such as interactive audio and video. Moreover, RTP includes payload type identification, sequence numbering, and

timestamping that are intended to ease the management of the transmission. However, RTP does not include protection mechanisms. Therefore, in order to avoid losses at a packet level, it is necessary to add complementary recovery modules.

Protection techniques are usually divided into two large families: ARQ schemes and FEC schemes. Whereas ARQ techniques are based on the retransmission of lost information, which is requested by the receiver, FEC techniques consist in generating redundancy data that are sent jointly with media data and can be used to recover lost information at the receiver.

Combination of both FEC codes and ARQ techniques has also been proposed in the hybrid automatic repeat request schemes (HARQ).

In time-sensitive communication, FEC-based schemes are usually preferred to ARQ schemes, since no-extra delay is added due to the retransmission of lost information. Nevertheless, FEC techniques also add some latency derived from the encoding and decoding operations.

In addition, although FEC-based schemes introduce additional band occupation, due to the transmission of redundant recovery data, they seem more suitable than ARQ techniques when it comes to time-sensitive multi-client communications. This occurs in ARQ schemes when each receiver suffer from different losses, so it has to send different requests to the transmitter, generating an additional traffic that may further contribute to the congestion of the network.

In this sense, protocols have been elaborated in order to provide a FEC-based protection scheme to RTP multimedia transmissions. The main idea is creating a parallel RTP-FEC packetized stream, composed of recovery packets, from a packetized media stream. This parallel stream is originated applying FEC codes across the RTP media packets. Each RTP-FEC packet contains an RTP header of its own, data that protect the RTP source headers, and the FEC payload. Different possibilities of FEC codes employed in this area are exclusive OR (XOR)-based codes, basic and interleaved, low-density parity-check (LDPC) codes, Reed-Solomon codes, fountain codes, etc.

The application and the optimization of FEC codes at the Applicaton Layer is an open research topic. In particular, this thesis only considers those FEC codes which are open source and deliberately leaves out patented codes (e.g., fountain codes). Moreover, they must meet different requirements in order to be used in time sensitive multimedia communication over IP networks. First of all, they must fit in a protection architecture based on the RTP protocol. Another important aspect is that they must be able to recover losses in bursty channels. Finally, the latency introduced by the code, during encoding and decoding operations, must be acceptable for the considered scenario. These three points leads the analysis and the proposed architectures presented in this thesis.

1.2 Objectives and thesis structure

The first objective of this thesis is the identification and the analysis of different types of FEC codes for packetized multimedia streams at the Application Layer. The analysis must take into account the characteristics of the considered scenario. The more relevant

characteristics are mainly two: the losses of the channel are in burst of packets, and the type of the multimedia data, protected by the FEC schemes, tollerates low latencies, since it is a time-sensitive transmission.

The second objective is the customization and optimization of those FEC codes that can make more reliable the time-sensitive communication for the considered scenario. This modification has to operate in order to strengthen those aspects of the code that are characterized by evident limitations.

Two well-known candidate codes for this thesis are Reed-Solomon and LDPC codes, since they are both no-patented codes and, in addition, they have interesting characteristics for the considered scenario. Reed-Solomon codes are extremely robust against multiple losses (either following a uniform distribution or a bursty behavior), whereas LDPC codes require a low computational cost, that permits lower encoding and decoding delays.

The analysis of Reed-Solomon codes addresses the main limitation of these FEC codes: their remarkable computational complexity. Thus, one of the objective of this thesis consists in reducing the computational cost by designing a novel architecture for the application of Reed-Solomon codes for protecting a media-packet stream.

The analysis of LDPC codes is centered on the study of their most remarkable limitation, that is, they are suited for memoryless channels, where losses occur following an uniform distribution of probability. Hence, another important objective of this thesis is the optimization of these codes by making them suitable to bursty channels and keeping their low computational cost. Moreover, the optimization has to take into account that LDPC codes are defined as large block codes, since their robustness increases with the number of information packets involved. Nevertheless, using a large number of information packets implies an increase in the latency, which may be inconvenient for real-time applications. Hence, the second main objective related to these codes is to reduce the size of the blocks, in order to maintain acceptable the required latency.

Some parts of this thesis have been published in [1] [2] [3] [4] [5] [6].

The rest of this thesis is organized as follows:

- **Chapter 2** presents a theoretical overview of FEC codes and it describes the main characteristics of Block Codes.

- **Chapter 3** introduces the use of FEC codes at the Application Layer.

- **Chapter 4** presents an alternative Forward Error Correction scheme, based on Reed-Solomon codes, the inter-packet symbol approach.

- **Chapter 5** presents a novel algorithm: the burst-oriented modifying algorithm. It allows boosting the recovery capability of LDPC codes against bursty packet losses.

- **Chapter 6** indicates the most important conclusions of this thesis and gives some proposals for future work that can extend the one presented here.

- **Appendix A:** presents an initial and partial evaluation of video response, in therms of distortion, for the packetized media stream protected by a small LDPC code.

It is also useful to remark that the last pages of this thesis contain a notation list describing the terms used throughout this thesis.

1.3 Main contributions of the thesis

The main contributions of this thesis are two and can be summarized as: (i) the definition of an architecture that optimizes the Reed-Solomon codes, from a computational point of view, and (ii) the design and optimization of LDPC codes based on a FEC architecture, from a recovery capability point of view, for Packet Erasure Channels, where erasures are generated as bursts of lost packets and low latency transmissions are requested.

Hence, on the one hand, we present an alternative approach to handle Reed-Solomon codes within an RTP protection scheme: the inter-packet symbol approach. This scheme, that is based on the well known interleaving technique, allows an alternative bit structure that allocates each symbol of the Reed-Solomon code in several RTP-media packets. The novelty consists in applying this technique to Reed-Solomon erasure codes for RTP-media streams. It permits to better exploit the recovery capability of Reed-Solomon codes against bursty packet losses. Our approach allows the use of lighter versions of Reed-Solomon codes compared to other solutions. These lighter versions results in a decrease of encoding/decoding time while keeping a comparable recovery capability.

On the other hand, in this thesis, we present an analysis to a priori evaluate the recovery capability of an LPDC code for a given IP network behaviour. The structure of an LDPC code is defined by a randomly generated parity matrix that identifies which packets are involved in the generation of each FEC packet. As this is a stochastic process, these matrices might not be uniformly robust, hence our approach is capable of assessing those parts of the parity matrix that are weaker against bursty losses and those parts that are stronger. Once the different parts of the parity matrix are outlined, in this thesis we propose a second algorithm to modify the weakest parts with the aim of improving overall recovery capabilities for channels with memory. Moreover, although our approach is based on giving a new structure to a randomly generated LDPC code, we preserve the intrinsic characteristics of the original LDPC code. This permits, at the receiver, to keep the same simple and original iterative decoding algorithm.

Chapter 2

Forward Error Correction Codes for Packetized Stream at the Application Layer

2.1 Introduction

Information streams are protected from errors and losses in communications channels by two main methods: one method is based on ARQ (Automatic Repeat-reQuest) protocol [7] [8] [9], the other method is based on FEC (Forward Error Correction) techniques [10] [11] [12] [13]. The principal difference between these two methods is that ARQ protocol can only detect errors and/or losses, but it is not able to correct them; hence, this protocol rejects the corrupted information and requests its re-transmission. On the contrary, FEC based techniques allow the detection and correction or recovery of errors and losses. The recovery capability of FEC based techniques depends on the number of redundancy symbols added to the original information stream.

The selection of one of these methods might be constrained by the delay that the communication system can tolerates. Obviously, a protocol based on ARQ needs more time than a protection FEC based system, since ARQ protocols involve several delaying operations: i) error detection, ii) request of the corrupted/lost information, and iii) wait of the re-transmission. This sequence of operations generates longer delays than the simple recovery, realized, in the case of FEC based schemes, at the receiver. Combination of both FEC codes and ARQ techniques has also been proposed as in the hybrid automatic repeat request (HARQ) [14].

Thus, in a system where only low delays can be admitted, such as video conferences, time-sensitive video contents, such as live broadcasting, etc., the issue of error and protection is usually solved by adopting FEC based techniques.

FEC techniques are based on channel codes [8] [10] [11] [12] [15] [16] [17]. The encoding consists in associating a group of information symbols (also called information vector) to a group of code symbols (code vector), for example by adding some redundancy symbols to the original vector. Hence, the application of a channel code to an

entire information stream is possible if the original flow is segmented in information vectors. Each of these vectors is encoded into code vectors, and a new encoded stream is transmitted.

There are two main families of FEC codes: Block Codes and Convolutional Codes. Typically, both types of codes are characterized by two parameters: n and k. The first one defines the number of the symbols of a code vector and the second one the number of symbols of an information vector. There is a third parameter, indicated by m, that represents the number of bits of a symbol. If it is not specified, it means that a symbol corresponds to a single bit.

The main difference between these two families of codes is that Convolutional Codes are memory codes, whereas Block Codes are memoryless [15] [17] [18]. This means that, for Convolutional Codes, the n symbols of a code vector do not only depend on the correspondent k information symbols, but also on some previous symbols. On the contrary, for Block Codes, the n symbols of a code vector are univocally dependent on the k symbols of the associated information vector. Hence, Block Codes have code vectors of a finite length, whereas Convolutional Codes do not and, moreover, they need complex algorithm such as the Viterbi algorithm to be decoded.

Usually, Block Codes are better suited for FEC schemes over packetized stream at the Application Layer than Convolutional Codes that are mostly used for lower layers of communication, in particular for physical layer and data link layer. This is due to the limitations introduced by the use of packetized stream architecture for multimedia communications. This type of protocols (e.g. Real-time Transport Protocol) requires two recognizable packet streams: one formed by source packets, with the data information, and another formed by recovery packets, with FEC redundancy [19] [20]. This means that each group of recovery packets must be associated to the group of source packets that has been used at the transmitter to generate them. This association is essential at the receiver, since, in the case of packets losses, the recovery operations can find immediately the relationship between source packets and recovery packets. This characteristics are easily identifiable in systematic Block Codes, whereas, for Convolutional Codes, the temporal dependency of previous received packets does not permit this distinction in the same straightforward way.

2.2 Block codes

A Block Code [13] [15] [18] is defined by three parameters, k, n and m:

- k, the length, number of symbols, of an information (or data) vector;
- n, the length, number of symbols, of a code vector;
- m, the number of bits of a symbol.

Another important parameter derived from k and n is the number of redundant symbols, and it is expressed as $r = n - k$.

A Block Code works in the following way: a binary stream is divided into information vectors of k symbols; each information vector is associated to a code vector of length n symbols. The new encoded sequence is transmitted through the channel. Figure 2.2.1 shows an example of an information stream protected by adding a parity bit to each 4 information bits. In this way a code, defined as $C(n = 5, k = 4, m = 1)$, or $C(n = 5, k = 4)$ when $m = 1$, is obtained.

...0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0... Original Information Stream

...0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0... Segmented Information Stream

...0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 0 0 1 0 1 0 1 0 0 1 1 0 0... Encoded Stream

Figure 2.2.1: Example of segmentation and encoding with a parity code C(5,4,1) for an information binary stream.

An important parameter is represented by the rate code, R_c, that is inversely proportional to the added redundancy:

$$R_c = \frac{k}{n} \tag{2.2.1}$$

This parameter also gives a measure of the latency that the code introduces: the lower R_c is, the higher the delay added by the FEC code is, since the quantity of redundant information ($n = k + r$) increases the time of transmission and, in case of losses, the decoding operations at the receiver.

2.2.1 Linear codes

In this thesis a concrete family of Block Codes is considered: the family of linear codes [11]. They are a particular class of block codes which are defined by a linear application (or linear function) between the Hamming space of dimension n and the Hamming space of dimension k. Given a set S of vectors of fixed length L, a Hamming space is a vector space over any set S. Typically, in coding theory, S is a Galois Field (GF) (or finite field). In the binary case, the Galois Field is $GF(2)$ (or $\mathbb{Z}_2$), and the Hamming space H_k is the set of all binary vectors with k components, as shown in 2.2.2:

$$H_k = \{\mathbf{v} = (u_1, \ldots, u_i, \ldots, u_k), u_i \, \epsilon \, \mathbb{Z}_2\} \tag{2.2.2}$$

In the following list three examples of a Hamming space over $\mathbb{Z}_2$ with $k = 1$, $k = 2$, $k = 3$ are shown:

- $H_1 = \{(0), (1)\}$;
- $H_2 = \{(00), (01), (10), (11)\}$;

- $H_3 = \{(000), (001), (010), (011), (100), (110), (101), (111)\}$

This means that H_k is formed by 2^k binary vectors. The sum and multiplication operations are defined within this space. The sum between vectors of H_k is obtained as an extension of the sum inside $\mathbb{Z}_2$, component by component. The results between single bits are shown in Table 2.2.1a. The multiplication is an operation within the Galois Field of dimension 2, $GF(2)$, and it is defined as a generalization, component by component, of the results shown in Table 2.2.1b. Thus, H_k is a vector space of dimension k in the $GF(2)$ in relation to these operations.

+	**0**	**1**
0	0	1
1	1	0

(a) Sum in $\mathbb{Z}_2$ between bits.

·	**0**	**1**
0	0	0
1	0	1

(b) Multiplication in $\mathbb{Z}_2$ between bits.

Table 2.2.1: Operations in $\mathbb{Z}_2$.

A vector space is univocally defined by a set of vectors, a standard basis, and for the vector space H_k is:

- $b_1(1, 0, 0, \ldots, 0)$;
- $b_2(0, 1, 0, \ldots, 0)$;
- $b_3(0, 0, 1, \ldots, 0)$;

  ...

- $b_k(0, 0, 0, \ldots, 1)$;

Vectors $\mathbf{v}$ and $\mathbf{c}$ are defined as the information vector and the code vector, respectively, and the mathematical notation is the following:

- $\mathbf{v} = (u_1, u_2, \ldots, u_k)\, \epsilon\, H_k$, information (or data) vector;
- $\mathbf{c} = (c_1, c_2, \ldots, c_k, \ldots, c_n)\, \epsilon\, H_n$, code vector;

Hence, we define the encoding operation as a linear function (or application, 2.2.3)

$$e: H_k \longrightarrow C \subseteq H_n \tag{2.2.3}$$

that associates each information vector $\mathbf{v} \epsilon H_k$ to each code vector $\mathbf{c} \epsilon C$, 2.2.4:

$$\mathbf{v} \longrightarrow \mathbf{c} \tag{2.2.4}$$

In this thesis we consider an encoding function, e, that meets the following properties that are very useful for encoding and decoding processes:

- **Systematic code**: the first k bits of a code vector correspond to the k bits of the information vector associated to it. Thus, the last $n-k$ bits of a code vector are exactly the parity vector $\mathbf{p}$ (see equation 2.2.5). This reduces the number of operations during the encoding and decoding process, since it is very easy to obtain the information vector directly from code vector that has generated it, and vice versa.

$$e(\mathbf{v}) = (\mathbf{v} \shortmid \mathbf{p}) \tag{2.2.5}$$

- **Linear code**: the code vector obtained by applying the transformation function e to two information vectors is equal to the sum of the transformations applied to the same two vectors, as (2.2.6) shows:

$$e(\mathbf{v_1} + \mathbf{v}) = e(\mathbf{v_1}) + e(\mathbf{v_2}) \tag{2.2.6}$$

This property is very useful to simplify the encoding operations since the $n-k$ parity bits of $\mathbf{c}$ can be obtained as a linear combination of the information bits of $\mathbf{v}$.

The encoding operation is an injective function. This means that each information vector generates a code vector that is different from the code vectors generated by the others information vectors. Hence, each received code vector is univocally associated to an information vector, as shown in the expression 2.2.7:

$$\forall\, \mathbf{v_1}, \mathbf{v_2} \epsilon H_k \; if \; e(\mathbf{v_1}) = e(\mathbf{v_2}) \Rightarrow \mathbf{v_1} = \mathbf{v_2} \tag{2.2.7}$$

2.2.2 Generator matrix G of systematic block codes

There is a matrix form that defines the generation of a vector code $\mathbf{c}$ from an information vector $\mathbf{v}$. For a given matrix G, called generator matrix of the code, vector codes are associated to information codes through:

$$\mathbf{c} = \mathbf{v}G \tag{2.2.8}$$

Matrix G is composed by the identity matrix I_k ($k \times k$ dimension) and by a parity matrix P ($k \times r$, where $r = n - k$), with coefficients a_{ij} equal to 0 or 1, depending on the type of the block code:

$$G = \begin{bmatrix} & & & k & & & r & \\ & 1 & 0 & 0 & ... & 0 & a_{11} & \cdots & a_{r1} \\ & 0 & 1 & 0 & \cdots & 0 & a_{12} & \cdots & a_{r2} \\ k & 0 & 0 & 1 & \cdots & 0 & \vdots & \ddots & \vdots \\ & \vdots & \vdots & \vdots & \ddots & \vdots & a_{1k}-1 & \cdots & a_{rk}-1 \\ & 0 & 0 & 0 & \cdots & 1 & a_{1k} & \cdots & a_{rk} \end{bmatrix}$$

This can be expressed as a system of linear equations with coefficients a_{ij} in $\mathbb{Z}_2$:

$$\begin{cases} c_1 = u_1; \\ \vdots \\ c_k = u_k; \\ c_{k+1} = a_{11}u_1 + \cdots + a_{1k}u_k; \\ \vdots \\ c_{k+r} = a_{r1}u_1 + \cdots a_{rk} \end{cases} \quad a_{ij}\,\epsilon\,\mathbb{Z}_2$$

A generic element g_{ij} of G is equal to 1 if the corresponding information bit u_i is involved in the relative equation in the system of liner equations. Otherwise it is equal to 0. In other words, the element g_{ij} of G designate which information bits u_i are employed to compute a bit c_j in the code vector.

2.2.2.1 Code properties and characteristics

There are some properties and characteristics of the codes that are useful to optimize the encoder and the decoder in terms of robustenss and complexity:

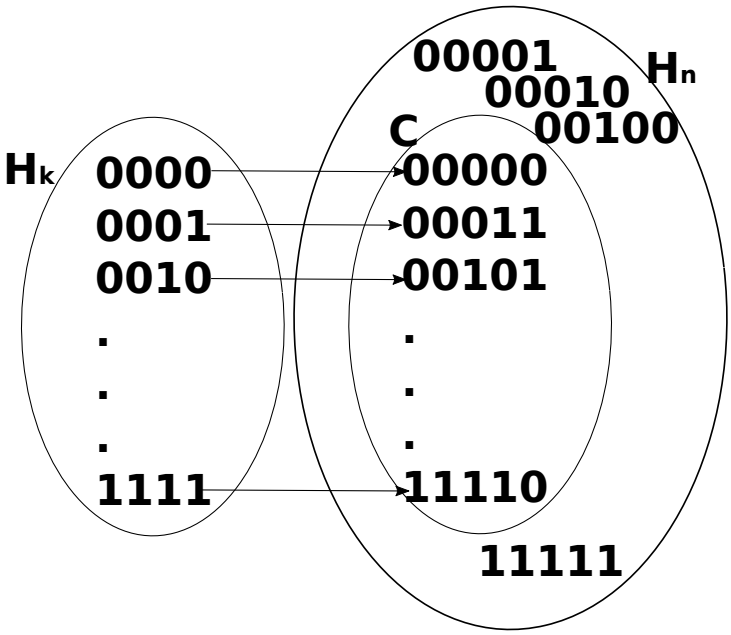


Figure 2.2.2: Code C as Image of the application $H_k \longrightarrow H_n$.

- The first main property is that any information vector $\mathbf{v}$ can be associated to any of all possible values of the code vectors within the vector space H_k.
- The encoder is a linear application between H_k and H_n. This means that the code C is an Image of the application, hence C is a subspace of H_n, Figure 2.2.2.
- The maximum rank of the generator matrix G is defined by the number of rows, since they are linearly independent. Thus the dimension of the vector sub-space generated is equal to the number of rows. This means that C is k-dimensional and it contains 2^k vector codes. It can be finally asserted that the k rows of G represent a basis of C.

For a given code vector, $\mathbf{c}$, of n elements, there are some parameters that characterize the code:

- *Hamming Weight*, which represents the numbers of 1s in a binary vector:

$$w_h(\mathbf{c}) = \{i : u_i = 1\} \tag{2.2.9}$$

- *Hamming Distance* between two vectors, that is the number of different elements between them:

$$d_H(\mathbf{c_1}, \mathbf{c_2}) = \{i : u_i \neq w_i\} \tag{2.2.10}$$

- *Minimum Hamming Distance,* which is the lowest value among the Hamming weights of all 2^kcode vectors. It is also used to represent the lowest value of all Hamming distances of a vector respect to different code vectors:

$$d_{min} = \min\ w_H(\mathbf{c}) \tag{2.2.11}$$

2.2.3 Decoding operation

There are two different ways to decode a received code vector [11] [18]:

- Hard Decoding: it works directly over the received binary vector;
- Soft Decoding: it works over the modulated signal, and it is based on the minimum energy criteria.

Since in this thesis we work at packet level, the Hard Decoding is the only possible decoding method in our scenario. It is also called minimum distance decoding: the decoding consists in comparing the received vector with all the existing code vectors. It can be demonstrated that this criteria minimizes the error probability over the set of vectors [10]. The decoding algorithm consists of several steps:

1. Calculate the Hamming distance between the received code vector and each of all the 2^k existing code vectors.
2. Choose the code vector that has the minimum distance from the received vector.
3. From this selection it is easy to recover the information vector, since it corresponds to the first k bits of the chosen code vector (systematic code).

As a simple example to understand the decoding by using a Hard decoding algorithm, the case of a FEC code with a Repetition code C(3,1) is presented. This simple type of code consists in repeating an information bit ($k = 1$) n times. Hence the generation matrix is:

$$G = [111]$$

The number of code vectors is $2^k = 2$:

$$C = \{\mathbf{c_0} = 000,\ \mathbf{c_1} = 111\}$$

Let the received vector be $\mathbf{y} = 101$. Then, the algorithm calculate the Hamming distance for each code vector:

- $d_H(\mathbf{y}, \mathbf{c}_0) = 2$;
- $d_H(\mathbf{y}, \mathbf{c_1}) = 1$;

Since $d_H(\mathbf{y}, \mathbf{c_1}) < d_H(\mathbf{y}, \mathbf{c_0})$, the algorithm chooses $\mathbf{c_1}$. Hence, considering that the code is systematic and the redundant bits are $n - k = 2$, the information vector is $\mathbf{v} = 1$.

Moreover, it could happen that there are more than one vector with minimum distance. In this case, the selection is randomly taken among the vectors that have the same minimum distance. For the sake of completeness, it has to be mentioned that the so-called class of *perfect codes* exists. The codes that belong to this class avoid the occurrence of multiple distance vectors, since they are always able to find a unique vector with minimum distance. They are Hamming codes, Golay codes (23,12), and repetition codes with odd n.

In the case of the exhaustive decoding, the computational cost is exponential, since it requests 2^k comparisons for each code vector. In order to reduce the computational complexity, several algorithms have been introduced. For example, algorithms based on standard array [21] or decoding through Trellis [11] [17] allow a minimum distance decoding with a computational cost proportional to 2^{n-k}. It is obvious that these algorithms still require a computational cost that can be unaffordable. For several practical application that needs sometimes very high values of $n - k$ and low latency, this computational cost is not acceptable. This is the reason why sub-optimal algorithms have been developed that require a lower computational cost. They are based on discrete mathematics techniques and some examples are represented by the algorithm of Berlekamp-Massey [18], used for the Reed-Solomon Codes [22] [23], or Chase algorithm [24], used in case of low error rate. In general, the type of decoding algorithm depends on the type of the FEC code employed and they are able to recover only error vectors with a low Hamming weight. Hence, it is not real Hard decoding, since a subset of recoverable error vectors are discarded, but they are very useful in practical cases when a limited decoding computational complexity is required.

In the specific case of erasure channels, since it is the case considered in the following chapters of this thesis, the position of the error, a loss, is known at the receiver. This obviously helps the decoding operations, since the comparisons of the Hamming distance is made only within a subset of the entire code vectors: the decoding algorithm only calculate Hamming distances for those code vectors that have the same no-lost symbols.

The computational cost analysis is considered more in detail in the chapters dealing with the use of Reed-Solomon and LDPC codes in a packetized communication scenario.

2.2.3.1 Block code recovery capability

The recovery capability is one of the most important characteristics of a FEC code, since, given a code with a *minimum hamming distance* d_{min}, see (2.2.11), it defines the number of errors that the code is able to correct for each code vector. This is an essential design parameter when a FEC code has to be chosen for protecting an information stream.

It is important to underline that vector codes, belonging to code $C(n, k)$ with minimum distance d_{min}, can recover all the error vectors within a limit Hamming weight, t, as shown in the following inequality:

$$t \leq \lfloor \frac{d_{min} - 1}{2} \rfloor \tag{2.2.12}$$

To demonstrate it, let us consider the case in which a code vector $\mathbf{c}$ is transmitted and the error generated by the channel is vector $\mathbf{e}$, with components equal to 1 in positions corresponding to an error and 0 if the bit is not corrupted. Hence, we can express the error vector $\mathbf{e}$ as a vector with Hamming weight lower or equal than t:

$$w_H(\mathbf{e}) = a \leq t \tag{2.2.13}$$

The received vector is:

$$\mathbf{y} = \mathbf{c} + \mathbf{e} \tag{2.2.14}$$

and the Hamming distance between $\mathbf{y}$ and $\mathbf{c}$ is:

$$d_H(\mathbf{y}, \mathbf{c}) = a \tag{2.2.15}$$

Let assume that the decoder is wrong choosing a vector $\mathbf{c_1} \neq \mathbf{c}$ and whose Hamming distance from $\mathbf{y}$ is lower or equal than a, in order that the following equation is verified:

$$d_H(\mathbf{y}, \mathbf{c_1}) = b \leq a \tag{2.2.16}$$

This event may happen with a probability $p > 0$ if $b = a$, $p = 1$ if $b < a$ (minimum distance decoding). This means that the Hamming distance between the vector $\mathbf{c}$ and $\mathbf{c_1}$ is:

$$d_H(\mathbf{c}, \mathbf{c_1}) \leq a + b \tag{2.2.17}$$

and it can be seen graphically in Figure 2.2.3, as the distance calculated between two points in a segment:

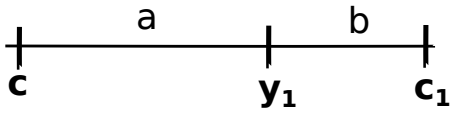


Figure 2.2.3: $d_H(\mathbf{c}, \mathbf{c_1}) \leq a + b$.

Since $b \leq a$, the expression (2.2.18) is true:

$$a + b \leq 2a \tag{2.2.18}$$

From equation 2.2.13, it can be written that:

$$2a \leq 2t \tag{2.2.19}$$

And we can assert that d_{min} is higher than $2t$:

$$d_H(\mathbf{c}, \mathbf{c_1}) \leq 2t < d_{min} \tag{2.2.20}$$

which is impossible, since the definition of minimum distance (see expression 2.2.11).

In the type of channels considered in this thesis, the erasure channels, where the position of the losses are known by the receiver, the recovery capability of the code is $2t$. Hence, the recovery capability of the block code is double respect to the case of correction with an unknown position of the error. This can be explained intuitively, since, half of the redundant symbols are needed to find the errors and the other half to correct them. If the position of the errors (erasures) is a priori known, then all the redundant symbols work to recover the erasures. The characterization of erasure channels will be discussed in detail in Section 3.3.

2.2.4 FEC codes at the application layer

Different possibilities of FEC codes employed in this area are eXclusive OR (XOR)-based codes, the basic scheme or interleaved one, low-density parity-check (LDPC) codes, Reed-Solomon codes, fountain codes, etc.

XOR-based codes are largely used in AL-FEC schemes, in particular the interleaved ones proposed by the Pro-MPEG forum in its Code of Practice (COP) 3 [25] and still employed at the Application Layer [26]. In this scheme, data vectors are organized in matrices, and recovery symbols are generated applying XOR operations by columns (1-D version), or, in some cases by rows as well (2-D version). In Figure 2.2.4 is shown a general example of this protection technique.

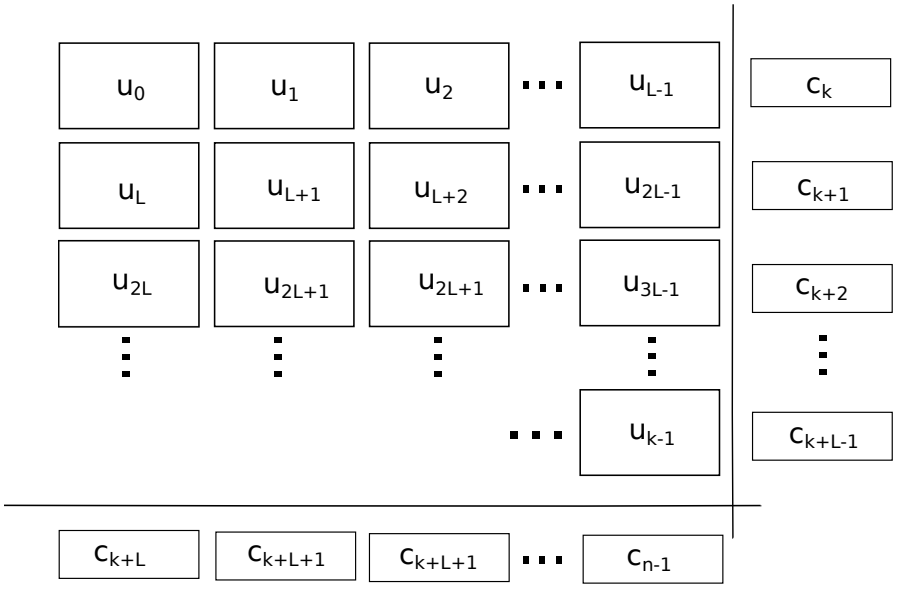


Figure 2.2.4: XOR Interleaving.

This type of scheme is particularly useful in the case of bursty losses, since recovery symbols have not been generated from consecutive data symbols. The recovery capability depends on the interleaving matrix dimensions: considering L as the number of columns, for the 1-Dimensional (1-D) version the recovery capability is L symbols, whereas, in the case of 2-Dimensional (2-D) version, the scheme is able to recover bursts up to $L + 1$. On the contrary, several isolated burst within an interleaving matrix are not recoverable if they occurs in the same column for the 1-D version, or in the same column and in the same row for the 2-D version. Hence, the main limitation of these FEC schemes is the rigid structure of the interleaving matrix for multiple single losses and for bursts longer than L. Since they have been largely studied and utilized in the packetized stream

environment, in this thesis they are used as reference in order to evaluate the recovery capability of our proposed schemes.

Reed-Solomon codes [22] [27] [28] [29] [29] [30] are patent-free codes and offer the best recovery capability for block codes, since they belong to the class of maximum distance separable (MDS) codes. In the case of an erasure channel, they are able to recover all the lost symbols of a data vector up to $n-k$ (the value of redundancy). This is the best recovery capability achievable by a block code. Moreover, they are able to recover bursty losses and also multiple isolated losses in the same data vector. Nevertheless, since they are codes that are based on polynomials over finite fields, they require a considerable computational cost for encoding and decoding operations. Their complexity can limit their use in real-time communications at the application layer [29] [31].

Another important family of FEC codes employed at the AL are LDPC codes [32] [33] [34] [35]. They can recover multiple (bursty or single) losses, as the data symbols, used to generate each redundant symbol, are not constrained in a fixed structure, as is the case of interleaved codes. The number of symbols employed in each equation that generates a redundancy symbol is very low. Moreover, the encoding operation is based on the XOR operation. These characteristics, XOR operations and a small number of data symbols involved in the generation of each redundancy symbols, require a lower computational cost with respect to Reed-Solomon codes. They can achieve a linear or quasi-linear encoding/decoding complexity [36] that is very useful in case of low latency communications. On the other hand, comparing LDPC codes with Reed-Solomon ones, LDPC codes are sub-optimal in terms of recovery capabilities [37], in particular against bursty losses.

Finally, a deeply studied family of FEC codes are Fountain and related codes [38] [39]. The most considerable difference from the Block Codes summarized in this Section is that they belong to the family of rateless codes, since they do not have a fixed rate code. This means that, theoretically, they can encode a finite information vector to an infinitely long code vector. Luby Transform (LT) codes are the first version of these codes but, due to their high encoding and decoding complexity, they have been improved to reach a linear encoding/decoding complexity. The evolution of LT-codes are represented by Raptors [40] [41] codes and Online codes [42] [43]. The encoding is based on simple XOR operations and the structure of the code is based on a bipartite graph, as LDPC codes. The main limitation of these codes is that they are patented, so they are not taken into account in this thesis.

2.3 Reed-Solomon codes

2.3.1 Introduction

One of the most important families of FEC codes are the Reed-Solomon codes. They offer the best recovery capability for block codes, since they belong to the class of maximum distance separable (MDS) codes. In that sense, they are extremely robust against multiple losses (either following an uniform distribution or a bursty behavior), whereas

the main limitation is that their application can be constrained by a remarkable computational complexity in the case that a large size of the Galois Fields is used.

It is very important to underline that the following Sections extend the theory exposed in previous sections, given other mathematical tools that permit to construct encoder and decoders based on Finite Fields symbols.

2.3.2 Finite fields

The theory of Finite Fields (or Galois Fields) is the fundamental mathematical language behind the Reed-Solomon codes [22]. The algebra related to Finite Fields defines operations (such as, sum and multiplication) that are basic to design the decoder and the encoder for a Reed-Solomon code. Hence, although the encoding/decoding operations follow the general rules of the linear block codes (see Section 2.2), it is essential a brief mathematical explanation of Finite Fields.

Two main representations of the elements of any Galois Field exist: the Exponential Representation and the Polynomial Representation. The first one is interesting to understand the multiplication within a Finite Field, whereas the second one is useful to apply the sum operations within a Finite Field. This Section is finalized to properly contextualize Primitive Polynomials (see Section 2.3.3), a class of polynomials that is the key of the encoding/decoding for Reed-Solomon codes.

2.3.2.1 Exponential representation: multiplication between elements of finite fields

For any prime number p, a finite field $GF(p)$ that contains p elements exists. It is possible to extend the elements of $GF(p)$ to p^m by generating the extended field called $GF(p^m)$, with m integer, positive, and greater than 0. The symbols within $GF(2^m)$ are used to construct the Reed-Solomon codes. The binary field $GF(2)$ is a sub-field of the extended field $GF(2^m)$. The elements of the extended field are indicated as 0,1 and α. It is possible to generate an infinite set of elements from the elements $\{0,1,\alpha\}$, by creating additional elements with the progressive multiply of the new elements by α, as indicated in 2.3.1:

$$F = \{0, 1, ..., \alpha, \alpha^2, ..., \alpha^j, ...\} \tag{2.3.1}$$

In order to obtain the finite field $GF(2^m)$ it is necessary to impose two conditions: (i) GF has to contain only 2^melements, (ii) it has to be a closed set for multiply and addition operations, that is, the results of these operations belong to the elements of the field. These conditions are represented by:

$$\alpha^{2m-1} = 1 = \alpha^0 \tag{2.3.2}$$

As a consequence of using this notation, the elements with an exponent greater than or equal to $2^m - 1$ can be converted to elements with exponents less than $2^m - 1$, as indicated in the following expression:

$$\alpha^{2m+n} = \alpha^{2m-1}\alpha^{n+1} = \alpha^{n+1} \tag{2.3.3}$$

Hence, the finite field $GF(2^m)$ is formed by the following elements:

$$GF(2^m) = \{0, \alpha^0, \alpha^1, ..., \alpha^{2m-2}\} \tag{2.3.4}$$

This type of exponential representation of the elements is very useful if the multiplication operation is needed among the elements of the field, since it is enough to add the indexes of the elements, considering the constraint given by:

$$\alpha^x \cdot \alpha^y = \alpha^{x+y} \tag{2.3.5}$$

2.3.2.2 Polynomial representation: addition between elements of finite fields

There is another mathematical representation for the elements of the field, that is called polynomial representation. This representation is useful for the sum operation. There is a correspondence between the elements of $GF(2^m)$, as they are indicated in (2.3.4), and the symbols of m bits in their binary representation. The generic expression that shows this correspondence is the following equation:

$$p(\alpha) = \sum_{j=0}^{m-1} \alpha^j \cdot i \tag{2.3.6}$$

with $i = 0$ if, in the binary representation, the bit in the correspondent j-exponent is equal to 0, whereas $i = 1$ if the bit is 1.

As an example, if we consider the case of $m = 3$, and the binary symbol "101", its polynomial representation is $\alpha^2 + 1$, whereas the polynomial representation of "111" is $\alpha^2 + \alpha + 1$.

Taking into account the different ways to represent the elements of $GF(2^m)$, a data vector or a code vector can be represented as a polynomial in X of degree $k-1$ in the case of a data vector, and of degree $n-1$ for a code vector. The expression (2.3.7) shows the way to represent a data vector $m(X)$ through a polynomial:

$$m(X) = \sum_{j=0}^{k-1} a_j X^i \tag{2.3.7}$$

The coefficients a_j are the elements of $GF(2^m)$ and the degree i of X indicates the position of a_j within the polynomial (code veector or data vector). Thus, data or code polynomial and data or code vector represent the same concept.

2.3.3 Primitive Polynomials

The objective of this Section is to make an overview over Primitive Polynomials that can be defined as the core of the encoding and decoding operation for the codes that works with Finite Fields. In particular, they are a very important class of polynomials, since they are used to generate the elements of the finite field $GF(2^m)$, switching from exponential notation to polynomial notation. Given an irreducible polynomial $f(\alpha)$ of

degree m, this is primitive if the lowest positive integer n, for which $f(\alpha)$ divides $\alpha^n - 1$, is $n = 2^m - 1$. This is a necessary and sufficient condition in order that a polynomial belongs to the class of primitive polynomials. Primitive polynomials are useful to convert the polynomial form to the exponential one: if we divide an element of $GF(2^m)$ in exponential notation by the primitive polynomial of degree m, we will have as the remainder the symbol in polynomial notation. In Table 2.3.1 there are some examples of primitive polynomials:

m	Primitive Polynomials
3	$1 + \alpha + \alpha^3$
4	$1 + \alpha + \alpha^4$
5	$1 + \alpha^2 + \alpha^5$
6	$1 + \alpha + \alpha^6$
7	$1 + \alpha^3 + \alpha^7$
8	$1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^8$
9	$1 + \alpha^4 + \alpha^9$
10	$1 + \alpha^3 + \alpha^{10}$
11	$1 + \alpha^2 + \alpha^{11}$
12	$1 + \alpha + \alpha^4 + \alpha^6 + \alpha^{12}$
13	$1 + \alpha + \alpha^3 + \alpha^4 + \alpha^{13}$

Table 2.3.1: Example of primitive polynomials.

The following example shows how to shift from a type of notation to the other. The parameters of the codes are:

- $m = 3$;
- $n = 2^m - 1 = 7$;
- $n - k = 2t$, (recovery capability in therms of symbols, for erasure case);
- $k = 2^m - 1 - 2t = 5$;

For $m = 3$ the primitive polynomial is $1+\alpha+\alpha^3$: the coefficients in polynomial notation are the remainders of the divisions of all the coefficients in exponential notation. In Table 2.3.2 there are all the elements of finite field $GF(2^3)$ with both representations.

Exponential notation	Polynomial notation	Binary value
0	0	000
α	α	001
α^2	α^2	100
α^3	$\alpha + 1$	011
α^4	$\alpha^2 + \alpha$	110
α^5	$\alpha^2 + \alpha + 1$	111
α^6	$\alpha^2 + 1$	101
α^7	1	111

Table 2.3.2: Elements of finite field $GF(2^3)$

Considering an example of a binary stream:

... 011 001 101 010 101 ...

that has to be represented by an information polynomial using the exponential notation, with symbol of $m = 3$, as shown in Figure 2.3.1.

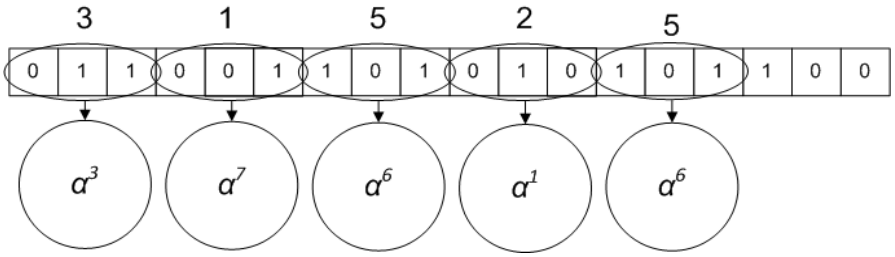


Figure 2.3.1: Binary stream to exponential notation.

The resulting expression is 2.3.8:

$$\alpha^3 X^4 + \alpha^7 X^3 + \alpha^6 X^2 + \alpha X + \alpha^6 \tag{2.3.8}$$

where the degree of X indicates the position of the coefficient within the information vector $\mathbf{v}$, where the elements are Finite Fields symbols, that is:

$$\mathbf{v} = (\alpha^3, \alpha^7, \alpha^6, \alpha, \alpha^6) \tag{2.3.9}$$

2.3.4 Reed-Solomon encoding

The typical way to represent a Reed-Solomon code, since it is a block code, is the following (2.3.10):

$$RS(n,k) = (2^m - 1, 2^m - 1 - 2t, m) \tag{2.3.10}$$

where $2t$ is the number of redundancy symbols and represents the recovery capability in the case of erasures.

There are two different ways to generate a Reed-Solomon code vector:

1. by multiplying an information polynomial by a generator polynomial;
2. by multiplying an information polynomial by a generator matrix;

2.3.4.1 Generator polynomial

The degree of the generator polynomial is equal to $2t$, that is, the redundancy of the code. Hence, there are $2t$ roots indicated as α , α^2, ..., α^{2t}. As a consequence of that, the way to represent the generator polynomial is given by the product of the sequence expressed in:

$$g(X) = \prod_{j=1}^{2t}(X - \alpha^j) \tag{2.3.11}$$

Once the generator polynomial is defined, the code polynomial is calculated as follows:

$$c(X) = g(X)v(X) \tag{2.3.12}$$

where $v(X)$ represents the information polynomial.

The main problem of this representation is that the result is a non-systematic code, that is, the first k symbols of the code vector do not correspond to the k symbols of the data vector. Reed-Solomon codes are cyclic codes, hence it is possible to reach a systematic codification by shifting $2t = n - k$ positions the data vector $v(X)$ of degree $k - 1$, transforming it into a polynomial of degree $k - 1 + 2t$. This shifting operation consists in:

1. multiplying the information polynomial by X^{n-k};
2. the new polynomial $X^{n-k}v(X)$ is divided by the generator polynomial $g(X)$;
3. the remainder is used as redundancy of the code vector. The following equation shows this in polynomial notation:

$$c(X) = p(X) + X^{n-k}v(X) \tag{2.3.13}$$

where $p(X)$ represents the remainder of the division expressed in:

$$p(X) = X^{n-k}v(X) \mod g(X) \tag{2.3.14}$$

2.3.4.2 Generator matrix

Since Reed-Solomon codes are linear codes, encoding can be done by multiplying a data vector by a $k \times n$ generator matrix (2.2.2) over a $GF(2^m)$, $\mathbf{c} = \mathbf{v}G$ (expression (2.2.8)).

A simple and efficient way to create a generator matrix G is based on using matrices that are commonly known as Vandermonde matrices, as indicated in the important work of Rizzo [29].

Let us consider a code with $n = 2^m - 1$, $0 < k \leq n$ and α as the root of the primitive polynomial of degree m chosen from the Table 2.3.1 for the corresponding value of m. The creation of a generator matrix for a Reed-Solomon code can be summarized as follows:

1. The construction of a non-systematic Vandermonde matrix $V_{k\times n}$, which (i, j)-entries are $\alpha^{i \cdot j}$, where α is the root of the primitive polynomial of degree m, $0 \leq i \leq k-1$ and $0 \leq j \leq n-1$. This type of matrix generates a non-systematic MDS code, since it transforms coefficients of a polynomial to the values that the polynomial takes at the point α. In other words, the Vandermonde matrix evaluates a polynomial at a set of points.

2. The inversion of the sub matrix $V_{k\times k}$, formed by the first k columns of $V_{k\times n}$. This is necessary to find a generator matrix for a systematic code.

3. The achievement of a systematic form for the generator Vandermonde matrix through the multiplication $V_{k\times k}^{-1} \times V_{k\times k} \times V_{k\times n}$.

In this way a systematic Vandermonde matrix is obtained, where the first k columns consist of the identity matrix I_k. This matrix can be used as the generator matrix of a Reed-Solomon code $C(n, k, m)$, and the final expression of G is shown in:

$$G = V_{k\times k}^{-1} \times V_{k\times k} \times V_{k\times n} = \left[I_k | V_{k\times(n-k)}\right] \tag{2.3.15}$$

The complexity of the pre-computation of the generator matrix can be estimated as the complexity of the multiplication of the inverse of a Vandermonde matrix by $n-k$ vectors (i.e., the last $n-k$ columns of $V_{k\times n}$) [30]. The complexity of the inverse of a $V_{k\times k}$ Vandermonde matrix by a vector is $\Gamma(k)_{inv}$:

$$\Gamma(k)_{inv} = \mathcal{O}(k \cdot (\log(k))^2) \tag{2.3.16}$$

hence, considering expression (2.3.15), the generator matrix can be computed in a number of operations proportional to $\Gamma(k, n)_G$.

$$\Gamma(k, n)_G = \mathcal{O}((n-k) \cdot k \cdot (\log(k))^2)) \tag{2.3.17}$$

Neverthles, creating G can be done off-line, so its complexity is not an issue.

The complexity of creating G, $\Gamma(k, n)_G$, is a value that depends only on off-line operations.

When the generator matrix is pre-computed, the encoding needs k operations per repair element (vector-matrix multiplication).

Encoding can also be performed by first computing the product of $\mathbf{v}$ by two Vandermonde matrices, as indicated in:

$$\mathbf{c} = \mathbf{v}V_{k\times k}^{-1} \times V_{k\times n} \tag{2.3.18}$$

The multiplication by the inverse of a square Vandermonde matrix is known as the interpolation problem and its complexity is $\Gamma(k)_{inter}$:

$$\Gamma(k)_{inter} = \mathcal{O}(k \cdot \log(k)^2) \tag{2.3.19}$$

The encoding can also be performed using mathematical tools (as Fast Fourier Transform), which reduce the total complexity of the encoding algorithm. Hence, the final expression for the complexity is indicated in:

$$\Gamma(k,n)_{enc} = \mathcal{O}((k/(n-k)) \cdot (\log(k)^2 + \log(k)) \tag{2.3.20}$$

2.3.5 Reed-Solomon decoding

In the scenario described in this thesis, the packets are received without errors or no received at all, that is known as the erasure channel case (see Section 3). This situation is crucial for the decoder, since, through the sequence number fields of the packet, it is able to know the position of the lost packets. In this case, the classic decoding algorithm is based on one of the most important property of the generation matrix: every sub-matrix of $k \times k$ dimension is an invertible matrix [30].

1. The first step consists in extracting the submatrix $k \times k$ dimensional obtained by the columns that corresponds to the received symbols. Moreover, since each encoded symbol is obtained by multiplying the information vector of k symbols by one column of the generator matrix, the received vector is the result of the multiplying operation by the submatrix of $k \times k$.

2. Since this matrix is invertible, the second step of the algorithm consists in inverting the submatrix formed by the vectors that correspond to the symbols of the received vector.

3. This matrix, multiplied by the received vector, permits to recover the lost symbols.

The complexity of this algorithm depends on the inversion of the submatrix and the multiplication vector-matrix. Using the Guass-Jordan algorithm, the inversion of the matrix requires $\mathcal{O}(k^3)$ operations, whereas the multiplying needs $\mathcal{O}(k^2)$ operations. As indicated in [44], if the value n is also taken into account, the decoding computational cost is proportional to $\Gamma(k,n)_{dec}$, as expressed in:

$$\Gamma(k,n)_{dec} = \mathcal{O}(k \cdot n) \tag{2.3.21}$$

In Table 2.3.3, the values of complexity for encoding and decoding for Reed-Solomon codes are summarized.

Encoding Complexity	**Decoding Complexity**
$\Gamma(k,n)_{enc} = \mathcal{O}(\frac{k}{n-k} \cdot \log(k))^2 + \log(k))$	$\Gamma(k,n)_{dec} = \log(k \cdot n)$

Table 2.3.3: Encoding/Decoding complexity for Reed-Solomon codes.

2.4 Low-Density Parity-Check codes

2.4.1 Introduction

Another important family of FEC codes, considered in this thesis, are the LDPC codes. They were introduced by Gallager [32]. However, they remained unused for more than 30 years, except for a small number of works, [45] [46] [47], until Mackay and Neal [33] [48], and, independently, Wiberg [49] rediscovered them. The relevance of this type of codes is related to the very low computational required complexity. This characteristic is obtained by (i) the use of XOR operations to generate the redundancy, and (ii) the low density of 1 entries in the generator matrix. Considering the first point, it can be asserted that XOR operations are less complex than the operations involved in encoding/decoding for Reed-Solomon codes [16] [29]. The second point is explained by highlighting that the number of 1 entries in a parity H matrix, and consequently in a generator G matrix, defines a low number of XOR operations for generating the redundancy.

They belong to the class of linear block codes, hence they are defined by parameters k, the number of symbols of a data vector $\mathbf{v}$, and n, the number of symbols of a code vector $\mathbf{c}$. Thus, the number of redundancy (or parity) bits is $n-k$, which, in the case of a systematic code, they are added to the k bits of a data vector.

As other linear block codes, an LDPC code is defined by its parity-check matrix H of dimensions $(n-k) \times n$, whose entries are exclusively 1's and 0's. A parity-check matrix H is defined by:

$$\mathbf{c}H^T = \mathbf{0} \tag{2.4.1}$$

The parity-check matrix is so named because it provides $n-k$ parity check equations that generate constraints between data symbols and parity symbols. Moreover, an LDPC code is defined as a linear block code for which the parity-check matrix H is very sparse, which means a low density (LD) of 1's. The methods to generate a parity matrix are multiple and in these representative works [33] [36] [50] several algorithms to construct H are presented.

2.4.2 Graph representation

The matrix H has an alternative representation: it can be expressed as a bipartite graph (also known as Tanner Graph) formed by left nodes (code nodes, corresponding to the symbols of a code vector) and right nodes (check nodes). A bipartite graph associates a parity-check equation to each check-node. Therefore, a check node (y_i) defines a parity-check equation formed by the code nodes (c_j) that are linked to y_i by edges. The number of the edges that come out from y_i and c_j defines the degree of the node.

From the parity matrix point of view, there is an edge that links a code node to a check node only if there is a 1 in the corresponding entry of H. The Figure 2.4.1 shows an example that illustrates a bipartite graph (with $n = 10$ and $n - k = 5$) and its corresponding parity matrix H. Notice that if the entry (i,j) of H is equal to 1, an edge links the check node y_i to the code node c_j.

Therefore, an LDPC code is defined univocally by its graph, which defines the set of code vectors that are involved in each parity check equation. All linear codes can be represented by a bipartite graph, however only the LDPC codes can be associated to a sparse bipartite graph.

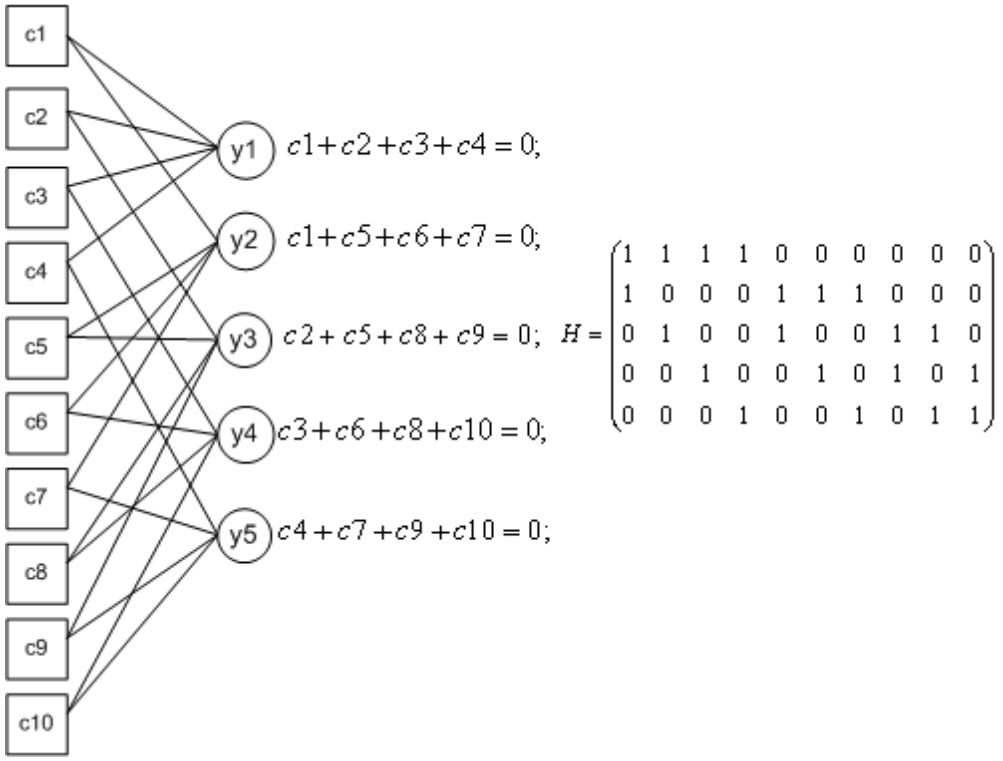


Figure 2.4.1: Bipartite Graph and the associated H matrix.

Another important characteristics are represented by the number of 1's in each column, w_c, and the number of 1's in each row, w_r. In addition, w_c indicates the degree of code nodes, and w_r corresponds to the degree of check nodes. Taking into account these parameters, an H matrix is defined as Low-Density when the conditions expressed in 2.4.2 and 2.4.3 are reached.

$$w_c \ll n - k \tag{2.4.2}$$

$$w_r \ll k \tag{2.4.3}$$

For a regular LDPC code the relation between the two parameters holds:

$$n \cdot w_c = (n - k) \cdot w_r \tag{2.4.4}$$

Otherwise, the LDPC code is called irregular [51]. In this thesis, only the regular version of these codes are considered, since a constant number of data packets that generate each FEC packet is a desirable occurrence.

Moreover, LDPC codes are defined as large block codes [52], that is to say, they perform better for high values of k. This characteristic depends on the distribution of 1 entries in H. In the case of erasure channels, a lost symbol cannot be recovered if two or more than two erasured symbols belong to the same parity-check equation. When the algorithm generates H, it fixes the number of 1 entries for each column (w_c), that is, the number of parity equations that contain the same symbol. As the value of k increases, the probability that two erasured symbols belong to the same equations decreases. This means that high values of k increase the probability that each sub-group of symbols used for a parity equation is different from the others equations.

2.4.3 LDPC codes encoding

As usual, in all linear block codes, in order to generate code vectors $\mathbf{c}$ from information vectors $\mathbf{v}$, the definition of generation matrix G is needed. G holds equation (2.2.8).

Thus, G is a $k \times n$ matrix, and it defines a unique correspondence from the space of vectors $\mathbf{v}$ to the sub-space of vectors $\mathbf{c}$ that fulfill the Parity-Check equations provided by H, in (2.4.1).

The main algorithms that create G from H consist in arranging H in an appropriate form that allows to develop G and construct it in a systematic form. Thus, H is randomly generated and then it is organized as:

$$H = [P^T | I_{n-k}]; \tag{2.4.5}$$

where I_{n-k} is the identity matrix of dimensions $(n-k) \times (n-k)$ and P is a sparse matrix of dimensions $k \times (n-k)$. So, the corresponding G matrix is:

$$G = [I_k | P]; \tag{2.4.6}$$

This approach is based on the use of the Gauss-Jordan elimination [34] [35] [36]. That algorithm, in general, has complexity of order:

$$\Gamma(k,n)_{G_{LDPC}} = \mathcal{O}(n-k)^3 \tag{2.4.7}$$

Depending on circumstances, the creation of matrix G from H can be performed offline. However, resulting matrix G in general is not sparse, owing to the Gauss-Jordan elimination process applied. This results in a straightfoward encoder implementation, that is quadratic in the block length. Hence, the encoding complexity is of order:

$$\Gamma(k,n)_{enc} = \mathcal{O}(n-k)^2 \tag{2.4.8}$$

with n around 1000 in large block coding.

It is possible to rearrange matrix H into an approximately lower-triangle form [36] so that it retains its sparseness, even after Gauss-Jordan elimination, because only some of the sub-matrices are affected. The order of encoding complexity then becomes $n + g^2$, where g is a small constant or scales as a small fraction of n [12]. The algorithm's software complexity does increase as a result, whatever the theoretical computational complexity.

2.4.4 LDPC codes decoding

As will be discussed more in details in Section 3.3, the type of the channels are the so called erasure channels. For this reason the dissertation over the decoding algorithms will be focused on this specific case.

In the case of erasure channels, the decoding operations are based on a simple iterative algorithm [53]: given a set of linear equations, if one of them has only one unknown variable, then its value is that of the constant term. This variable is replaced in all remaining linear equations, and the algorithm reiterates. Hence, several unknown variables can be found by the recursive algorithm.

In Figure 2.4.2 it is shown an example for a very simple case of decoding using an iterative algorithm for an erasure channel.

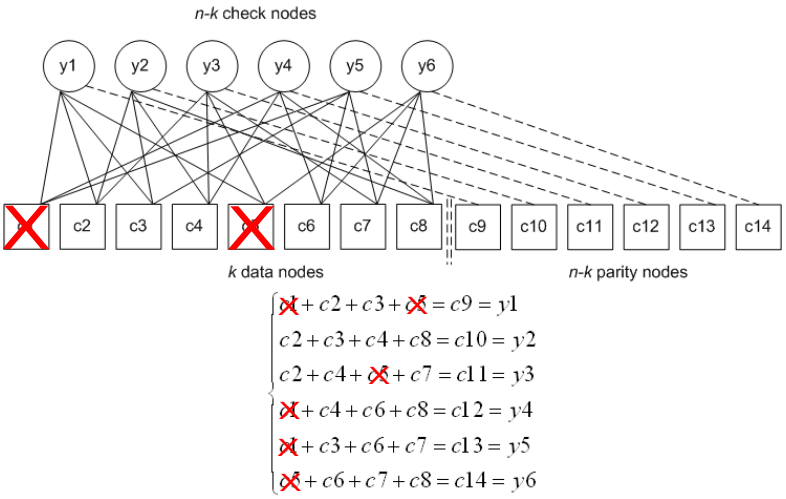


Figure 2.4.2: Example of iterative decoding in LDPC codes.

In the example there is a code vector, with $n = 14$ and $k = 8$, which has suffered two losses, symbol c_1 and c_5. Moreover, in Figure 2.4.2, the bipartite graph used to generate the $n - k = 6$ parity symbols is presented. The decoding algorithm main steps can be summarized as follows:

1. The decoder detects the losses, c_1 and c_5.
2. The algorithm searches for the first equation where c_1 is situated, which is the first one;
3. It verifies if there are more than one lost symbol involved in this equation;
4. Since c_5 is situated in this equation, the algorithm searches for another equation where c_1 has been used to generate a redundancy symbol;
5. This situation occurs in the fourth equation, hence, using the data symbols (c_4, c_6, and c_8) and the redundant symbol (c_{12}), c_1 is recoverable.
6. The algorithm search for the first equation where c_5 has been used, which is the first one;
7. This time, since c_1 has been recovered, thanks to the fourth equation, c_5 is also recoverable.

Several works show how the iterative decoding, when the decorder are carefully desinged, by cascade for examples ([54] and [55]), or as shown in [56], achieves a linear complexity.

2.4.5 LDPC codes families

Several families of LDPC codes exist. The differences between the families of LDPC codes depend on the characteristics of the parity matrix. These characteristics have been developed in order to optimize the encoding operations and, consequently, the decoding operations.

The types of LDPC codes considered in this Section are a simplification of the general case. The simplification consists in reducing the complexity of the pattern of matrix H. The family considered are mainly three: Low-Dense Generator-Matrix codes (LDGM), staircase LDPC, and triangle LDPC. They are enumerated from the most simplified to the most elaborated version. Moreover, the recovery capability is inversely proportional to the simplicity of the code.

2.4.5.1 Staircase and Triangle LDPC codes

Two deeply studied types of LDPC codes are staircase and triangle LDPC codes [50] [57] [58].

In order to define these two types of LDPC codes, the parity matrix H is considered divided into two parts:

1. the left part (from column 0 to column $k-1$, that corresponds to the information symbols) H_1;

2. the right part of the matrix (from column k to column $n-1$, corresponding to parity symbols), H_2;

$$H = [H_1|H_2]$$

The only difference between this two codes is the generation of the right sub-matrix, H_2. The staircase version $H_2^{staircase}$ is a bi-diagonal matrix:

$$H_2^{staircase} = \begin{bmatrix} 1 & 0 & ... & 0 \\ 1 & 1 & ... & 0 \\ ... & ... & ... & 0 \\ 0 & ... & 1 & 1 \end{bmatrix}$$

whereas for triangle version $H_2^{triangle}$ is a triangular matrix:

$$H_2^{triangle} = \begin{bmatrix} 1 & 1 & ... & 1 \\ 0 & 1 & ... & 1 \\ ... & ... & ... & 1 \\ 0 & ... & 0 & 1 \end{bmatrix}$$

For this two types of LDPC codes, there is a very trivial iterative decoding algorithm, introduced by Zyablov and Pinsker [45]. There is an improved hybrid version of this algorithm [59], based on the scheme of Gaussian elimination.

2.4.5.2 LDGM codes

The basic difference between classical LDPC codes and LDGM codes is that each control node is connected to only one parity node. This means that the parity-check matrix, which dimension is $(n-k)\times k$, only takes into account the first k nodes, and is associated to the identity matrix of dimensions $(n-k)\times(n-k)$. Hence, each parity node is connected to only one control node as shown in Figure 2.4.3:

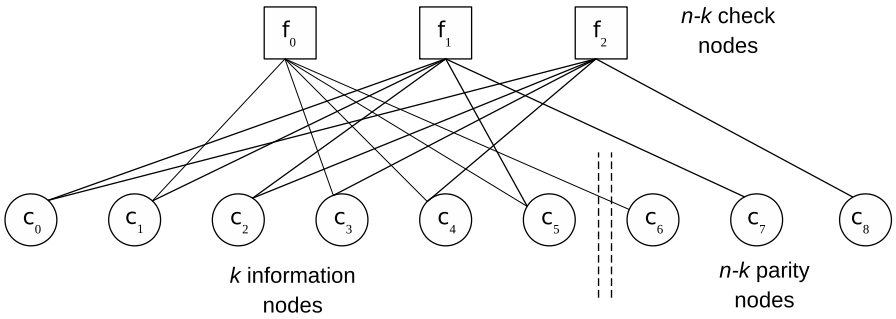


Figure 2.4.3: Bipartite Graph for a LDGM code.

This codes generates a system of parity-check equations, associated to the graph in Figure 2.4.3, that is:

$$\begin{cases} f_0 : c_1 + c_3 + c_4 + c_5 + c_6 = 0 \\ f_1 : c_0 + c_1 + c_2 + c_5 + c_7 = 0 \\ f_2 : c_0 + c_2 + c_3 + c_4 + c_8 = 0 \end{cases}$$

and the parity-check matrix is the following:

$$[H_1|I_{3\times 3}] = \left[\begin{array}{cccccc|ccc} 0 & 1 & 0 & 1 & 1 & 1| & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1| & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0| & 0 & 0 & 1 \end{array}\right]$$

This characteristic, each parity node is connected to only one control node, produces differences between LDPC codes and LDGM codes, from the encoding speed and recovery capability. LDGM codes, in fact, need less encoding time than LDPC codes. Nevertheless, LDPC codes obtain better results if they are compared to the recovery capability results.

As the case of classic LDPC codes, LDGM codes are regular codes too. The equation that relates the parameter n and k to the degree of control nodes and information nodes in the bipartite graph, is different from the LDPC case (expression (2.4.4)), and is expressed by:

$$k \cdot w_c = (n - k) \cdot w_r \tag{2.4.9}$$

The parity check matrix is randomly generated, taking into account the limitation given by the (2.4.9).

The encoding, in the case of LDGM codes, is very simple: the encoder generates each protection symbol applying the XOR operations to all the information symbols associated to the check node corresponding to the parity node. As a consequence, the generator matrix is directed obtained from parity matrix.

A typical decoding algorithm is based on the solution of a linear equation system, by an iteration algorithm.

This family of LDPC codes is particularly important for this thesis, since, the algorithm of modification of the parity matrix exposed in Chapter 5, that can be considered one of the most important contribution of this work is mainly designed around, but not only, the LDGM codes.

Chapter 3

FEC at the Application Layer

3.1 Packet erasure channels

The scenario, where the FEC architectures defined in thesis are placed is the transmission of data through a packetized stream, since, in the last years, multimedia communications are moving toward IP [60] [61] [62]. In particular, the data considered in this work is video for real-time communications or time-sensitive applications.

In this type of communication systems, packets suffer from losses depending on several causes, and, in the literature, the channels with the characteristics previously presented, are usually defined as Packet Erasure Channels (PEC). In PEC, packets are received correctly or are lost. Hence, a sequence number value is typically associated to each packet, and it permits, at the receiver, to know when a packet has been lost. The main cases of losses are summarized in three points:

1. Some part of the packet is corrupted, generally by the channel noise. It can be established by other protection mechanism at different level, such as Cyclic Redundant Check (CRC) or Parity Check bits. In this case the packet is considered lost and, as a consequence, discarded.

2. The packet arrives at the receiver, but the delay is higher than the allowed limit for the type of the communication. Hence, the mechanism of time out is activated and the packet is considered lost.

3. Some packets, depending on the congestion of the net, can be randomly discarded by the routers.

From the point of view of the coding theory, in a PEC, the knowledge of the position of the erasures permits the FEC code to achieve a recovery capability that is double respect the generic case (see Section 2.2.3.1). The intuitive explanation is the following: since, half of the redundant recovery equations are needed to find the errors (for instance, parity-check equations) and the other half to correct them, if the position of the errors (erasures) is a priori known, all the recovery equations, and consequently redundant symbols, work to recover the erasures.

A classical example is given by a FEC code based only on the simple XOR operation over k data symbols. The redundancy symbol generated by the XOR is only useful to determinate the presence of an error or not. For this reason, it is also known as parity symbol. Nevertheless, if the position of the error, or the loss (erasure case), can be deduced thanks to some type of mechanism (i.e. Sequence Number), the decoder is able to reconstruct it through the reverse application of the XOR operation to the $k-1$ received data symbols plus the redundancy symbol.

This can be easily extended to the packetized streaming case: let us consider a group of data packets that have suffered from one single lost and the protection scheme employed is the well-known XOR-interleaved matrix [25] [26] [63], as shown in Figure 3.1.1. The FEC-packet generated from the data packets in the first rows permits to know only the presence of a loss. The protection system needs the FEC-packet generated from the first column to find the position of the lost data packet and to recover it. If the position of the lost is a priori known by the decoder, the FEC-packet of the first row would be enough to reconstruct it.

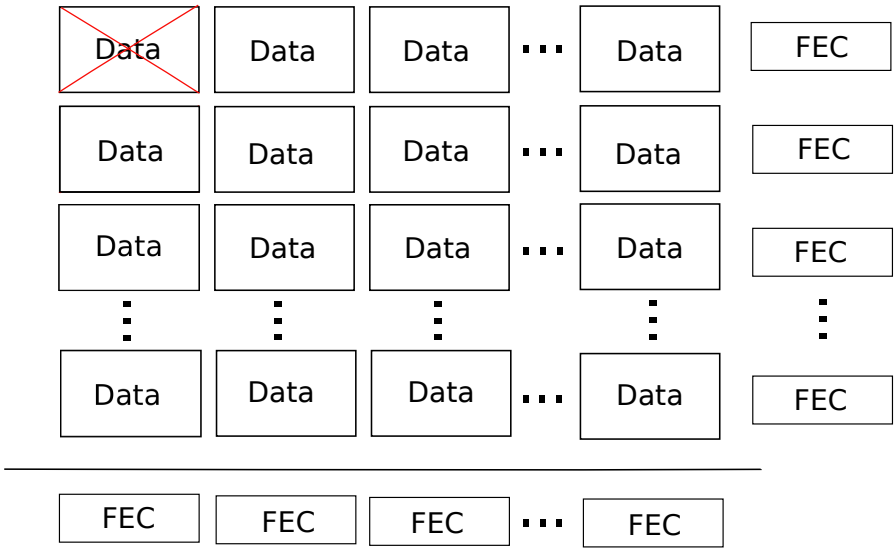


Figure 3.1.1: Interleaving Example For Packet Erasure Channel.

3.2 Real-time multimedia communication in IP channels

Although the FEC schemes proposed in this thesis are totally independent of the protocols employed, in order to use a communication system that could be compared to the state of the art, the following chapters have to be considered in the context of IP/UDP/RTP combination (see Figure 3.2.1) .

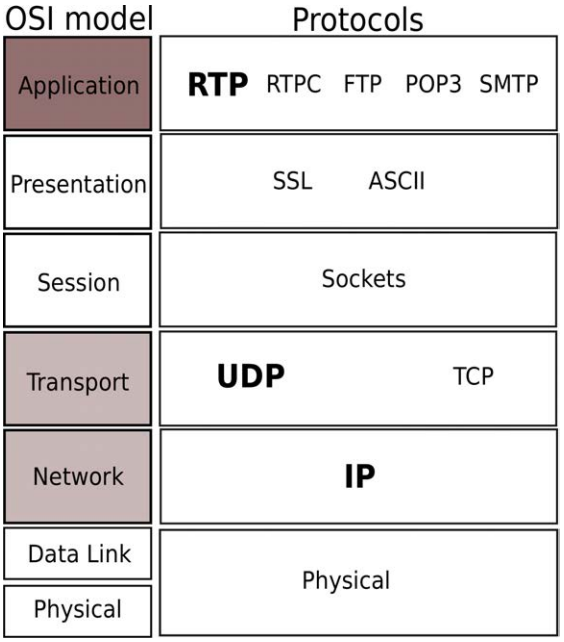


Figure 3.2.1: OSI Model: IP/UDP/RTP combination.

In particular, protection schemes operate at the application layer, and they follow, as general reference, the architecture presented in the FEC framework [20]. This election is motivated by the unreliable nature of IP networks. Hence, IP is usually employed with the transport level protocols TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP provides reliable and ordered packet delivery and it is employed by the most popular internet applications, such as World Wide Web (WWW), E-Mail, etc. UDP is commonly used in time sensitive application, since it permits lower delays thanks to a simple transmission model. Nevertheless, UDP provides an unreliable communication: neither lost packets retransmission nor the order of the received packets are guaranteed.

Hence, it is necessary to include mechanisms in order to give reliability to the communication: a typical solution is partly represented by the Real-time Transport Protocol (RTP), an application layer protocol developed by the Video Audio Transport Working Group of the Internet Engineering Task Force (IETF) (RFC 3550 [64]) that is encapsulated in UDP. RTP defines a packet format for delivery data with real-time characteristics, such as interactive audio and video. Moreover, RTP includes payload type identification,

sequence numbering and timestamping that are intended to ease the management of the transmission.

However, RTP does not include protection mechanisms, therefore, in order to avoid losses at packet level, it is necessary to add complementary recovery modules. In this sense, the Video Audio Transport Working Group of IETF has elaborated RFC 5109 [19] in order to provide a FEC-based protection scheme to RTP multimedia transmissions. RFC 5109 recommends creating a parallel RTP-FEC packetized stream, composed of recovery packets, from a packetized media stream. This parallel stream is originated applying FEC codes to the RTP media stream (RFC 3550).

3.3 An overview of the Real-time Transmission Protocol

RTP [64] defines a packet format for delivery data with real-time characteristics, such as interactive audio and video. The RTP-payload is the data transported by a packet, that in the case considered in this thesis is basically represented by compressed video data. Moreover, RTP includes an RTP-header of a fixed size, that is the first 12 bytes in the standard case.

The RTP-header has important fields, such as payload type identification, sequence numbering and timestamping that are intended to ease the management of the transmission.

The main RTP-header fields are shown in Figure 3.3.1.

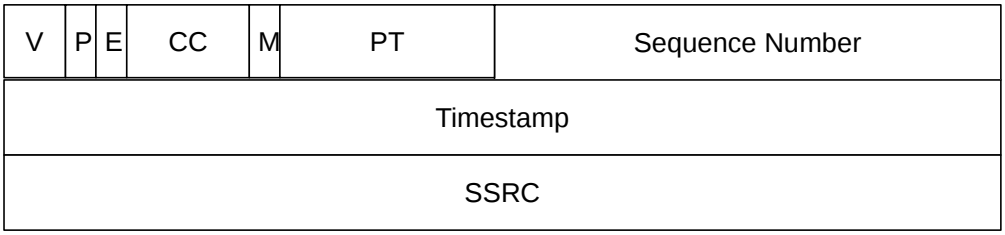


Figure 3.3.1: RTP packet Header.

The details for each fields of the RTP header can be found in RFC 3550. In the following list they are summarized thoroughly for the purposes of this thesis:

- **Version (V)**: 2 bits, it indicates the protocol version. It is set to 2, since this thesis follows the recommendations.
- **Padding (P)**: 1 bit, it indicates if the padding has been used in the RTP-media packet. The padding consists in one or more additional padding octets at the end of the payload that does not contain data information. The last octet of the padding contains the number of the octets should be ignored, itself included. Padding is necessary when a packet fixed length is needed.

- **Extension (X)**: 1 bit, in an RTP-media packet indicates if the packet has extension fields.

- **CSRC count (CC)**: 4 bits, it contains the CSRC-list identification. This is the list that indicates the source of the payloads of the packet.

- **Marker (M)**: 1 bit, it allows to indicates particular events in the packet stream, for example some important frame contained in it.

- **Payload type (PT)**: 7 bits, it identifies the format of the RTP-packet payload. For example audio or video, or FEC/RTP packet. The receiver has to be able to discard packets with an unknown payload type.

- **Sequence Number (SN)**: 16 bits, this value is refreshed when an RTP-media packet is sent. It is useful at the receiver, since, through it, it is possible to detect losses in the packet stream or to recover the original order of the packets.

- **Timestamp**: 32 bits, it represents the time-sample of the first RTP byte. It is set to the value of the RTP-clock in the time when the RTP-media packet is sent.

- **SSRC**: 32 bits, this field identifies the synchronization source. This identifier should be randomly chosen, in order to have two different SSRC identifier for two different sources in the same RTP session.

There are more fields that can be added by using the extension, but in this thesis they are not considered.

However, RTP does not include protection mechanisms, therefore, in order to avoid losses at packet level, it is necessary to add complementary recovery modules. In this sense the Internet Engineering Task Force (IETF) has spent several years working on RFCs (Request For Comments) that propose standards and recommendations about FEC techniques and recovery architecture applied to RTP protocol streams.

3.4 FEC-RTP protocol

3.4.1 FEC framework

The protection system employed in this thesis works at the Application Layer, and it follows the architecture presented in the FEC framework RFC 6363 [20]. This means, basically, that two RTP instances are generated: (i) one for the source packets and (ii) another one for the repair packets (RTP-FEC packets). Thus, an RTP-FEC packet contains an RTP header of its own, and the redundancy data for the RTP source header and its payload. The redundancy to protect header and payload is generated by applying the FEC code across the RTP source packets.

In Figure 3.4.1, the general architecture where the protection schemes proposed in this thesis are situated, is shown. As indicated before, two RTP instances are used: one instance is for the media data and another one is for the recovery information. This is because the use of RTP for the media data must be separated from, and independent of, the use of RTP for the FEC packets. In this thesis, the existence of two RTP instances is designed as two separated streams. This is a reasonable choice when, in block FEC codes, the repair payload contains redundancy data encoded across the RTP headers of the data packets. Thus, a redundancy packet carried over RTP starts with an RTP header of its own, which is followed by the redundancy data containing bytes that protect the source RTP-media headers and the RTP-media payloads.

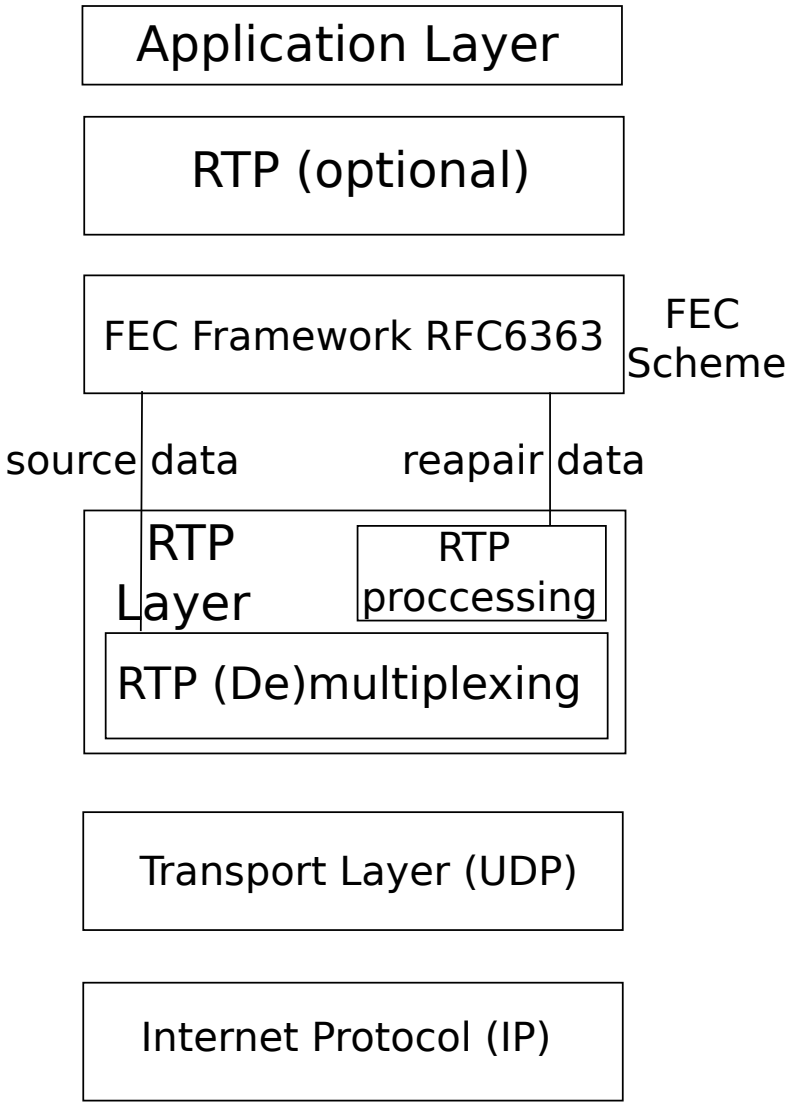


Figure 3.4.1: FEC Framework Architecture RFC6363.

In the specific case of this thesis, in order to generate the FEC/RTP packet fields, we have followed RFC 5109 [19], that recommends some basic rules about the structure of the protecting packet. It says how to fill the RTP header field, the FEC/RTP and payload. Each FEC/RTP is generated from a fixed number of RTP-media packets by using a generic FEC code. In Figure 3.4.2 the different streams are shown, the information stream segmented in groups of k packets, and the protection stream, segmented in groups of $n-k$ packets. In Section 4.2.1 is shown a practical implementation of how to generate the recovery fields for a FEC/RTP packet from the field of the RTP packets that the architecture has to protect.

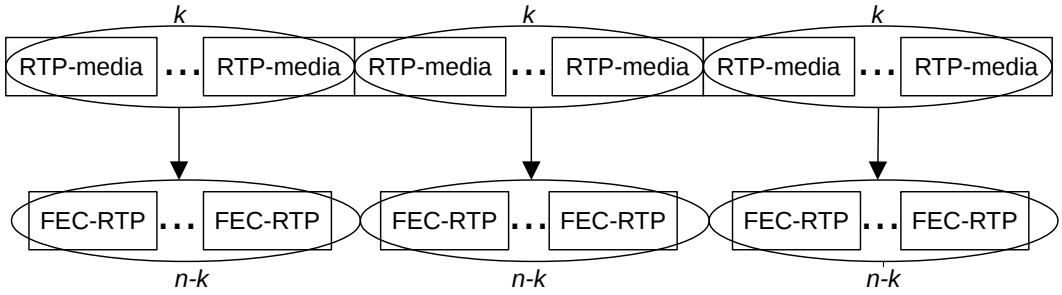


Figure 3.4.2: Generation of a FEC/RTP stream from a RTP-media stream.

Moreover, RFC 5109 explains how to distinguish the recovery packets from the media packets, and how to know from which RTP-media packets a FEC-packet has been generated. This is essential at the receiver, in the case of recovery of lost packets. The generation of two different RTP instances, one for the source packets and another one for the repair packets, is very useful for the communication systems that implements FEC mechanisms, in order to differentiate the two flows, and for those that do not it, in order to discard FEC packets.

3.4.2 FEC/RTP packet structure

A FEC/RTP packet has a similar structure to an RTP-media packet, although there are some other important fields for the recovery of lost packets.

It is constituted, in its basic form, by an RTP header (12 Bytes), FEC Header (12 bytes), and FEC payload. In Figure 3.4.3 is shown a FEC/RTP packet generic structure.

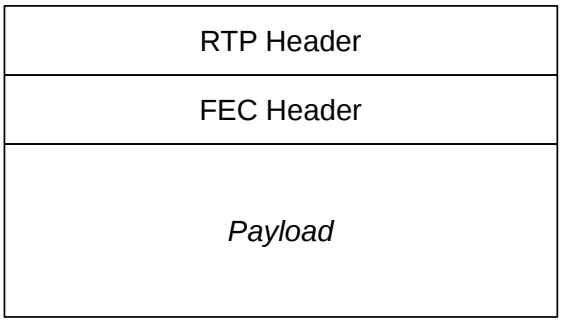


Figure 3.4.3: FEC/RTP packet structure.

The first difference, with respect to an RTP-media packet, is the existence of a FEC/RTP header, that contains the fields used for establishing which RTP-media packets has been used to generate it.

3.4.2.1 RTP header of a FEC/RTP packet

Figure 3.3.1 shows the structure of the RTP header of a FEC/RTP packet, that is the same of an RTP-media packet.

The Field of the RTP header are summarized in the following list.

- **Version (V)**: 2 bits, it indicates the protocol version. Usually it is set to 2 in binary value (10).
- **Padding (P)**: 1 bit, in the case of a FEC/RTP packet, it is the result of the protection operation applied to the Padding fields of the RTP-media packets used to generate the considered FEC/RTP packet.
- **Extension (X)**: 1 bit, it is the result of the protection operation on the same field of the k RTP-media packets. In an RTP-media packet indicates if the packet has extension fields.
- **CSRC count (CC)**: 4 bits, in the case of a FEC packet, it derives from the encoding operation over the same field of the k RTP-media packet associated to it.
- **Marker (M)**: 1 bit, it is the result of the encoding operation on the same filed of the k RTP-media packets.
- **Payload type (PT)**: 7 bits, in the case of a FEC/RTP, it can be chosen as a different value respect to the payload type field of a source packet. With this method, it is easier to distinguish which packets are FEC/RTP packets, and discard them if recovery operations are not supported.

- **Sequence Number (SN)**: 16 bits, this value is refreshed when an RTP-media packet is sent.
- **Timestamp**: 32 bits, it is set to the value of the RTP-clock in the time when the FEC/RTP packet is sent. It is updated independently of the employed FEC scheme.
- **SSRC**: 32 bits, in the RTP-header of a FEC/RTP, it contains the same value of the data stream that the FEC scheme protects. It can be different in the case a multiplexed FEC-RTP/Media-RTP is generated. In this way the receiver is able to recognize a FEC packet by the SSRC identifier, instead of using the Payload Type field.

These are the basic fields in our protection architecture. There are more fields that can be added by using the extension, in the case the application needs them.

3.4.2.2 FEC header of a FEC-RTP packet

Figure 3.4.4 shows the fields of a FEC header of a FEC-RTP packet:

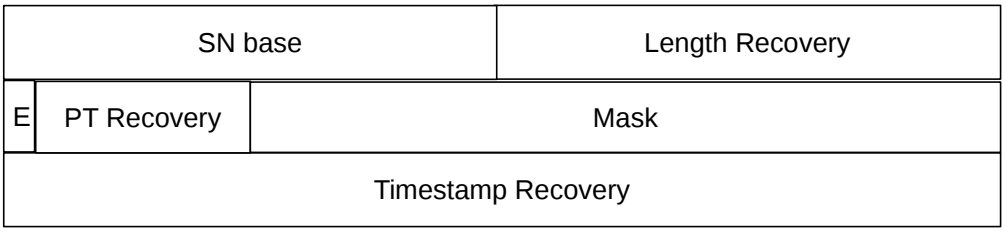


Figure 3.4.4: FEC header.

The FEC header is constituted by 12 bytes that are organized in the following fields:

- **Sequence Number (SN) base**: 16 bits, it is set to the lowest value among the sequence numbers of the RTP-media packets protected by the FEC/RTP packet. The maximum number k of RTP-media packets, which can be protected in the same group, depends on the length of the field Mask. It is set to 24, but can be modified depending on the employed protection scheme, as indicated in the example proposed in the field Mask.
- **Length recovery**: 16 bits, it is a field used to recover the lost RTP-media packet length, since it is the result of the protection operation applied to the length values of every k RTP-media packets. It is a very important field, since it permits to know, once the payload of a lost RTP-media packet is recovered, how many bytes belong to the information data and how many bytes belong to the padding. The padding is a necessary operation in order to have the same length for all the k RTP-media packet payloads, since RTP-media packets can have different sizes.

- **E**: 1bit, always set to 0.
- **Payload Type (PT) recovery**: 7 bits, this field is obtained by applying the protection operation to the bits of the Payload Type field in the protected RTP-media packets.
- **Mask**: 24 bits, the recommendation says that if the i-th bit is equal to 1, then the RTP-media packet marked by the Sequence Number $N + i$, is associated to this FEC/RTP packet, where N indicates the value of the Sequence Number base. The less significant bit is $i = 0$, whereas the most significant is $i = 23$. For this reason the highest number of packet group can be 24. For some protection schemes this number is too low, so the Mask field is used in a different way. For example in the case of an interleaved XOR, this field can be employed to indicate the number of columns and rows for the interleaving matrix [65].
- **Timestamp Recovery**: 32 bits, it is calculated through the protection operation over the Timestamp fields of the packets to protect.
- **Payload FEC**, the RFC indicates that it has to be the result of the encoding operations applied to some concatenating fields as CSRC-list, RTP-extension, the media payload and padding of the RTP-media packets associated to this FEC/RTP packet.

3.4.3 Protection operation

Although there is not a single solution to protect a group of k RTP-media packets, following RFC 5901, a very practical and didactic way is proposed in this Section. Taken a group of k data packets, for each packet a string is generated. This string is called information or data, and it is the result of concatenating the RTP-header fields plus the payload that are protected during the encoding process:

- Padding Bit (1 bit);
- Extension Bit (1 bit);
- CC bits (4 bits); Marker bit (1 bit);
- Payload Type (7 bits);
- Timestamp (32 bits);
- The Payload of the packet and the padding;
- The length recovery of the packet;

As discussed before, the padding is necessary to have k-payloads of the same length, and the length recovery remind the real number of data byte for that packet.

Afterward, the strings have to be considered as the rows of a matrix, and the FEC code is applied to the byte of each data string by columns. This will generate n-redundancy bytes in n-rows, that is, the FEC strings. Finally from the FEC strings, following RFC 5109 (see Section 3.4.2.2 and 3.4.2.1), the FEC/RTP packets are composed.

In order to have a general idea about how to apply the FEC codes in a practical way to RTP-media packet fields, it can be useful to consider the example proposed in Figure 3.4.5. It summarize graphically how to apply a FEC code $C(7, 4)$.

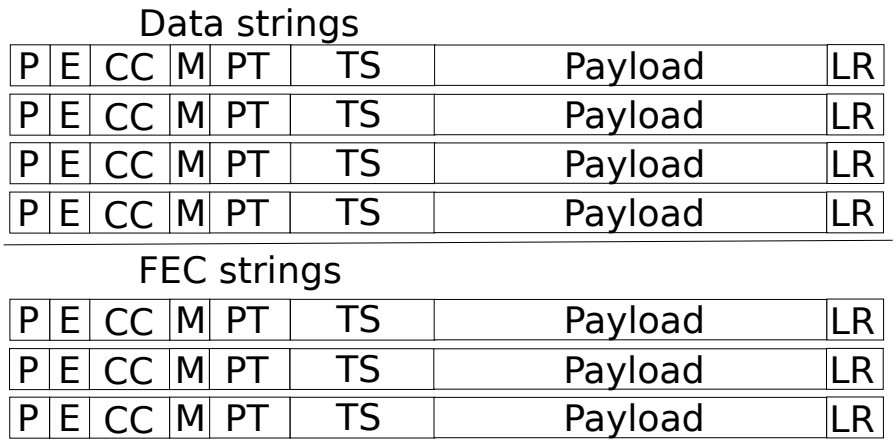


Figure 3.4.5: FEC strings generated from Data strings.

The operation of generating the string is repeated for all the $k = 4$ media packets. The generic FEC code $C(7, 4)$ is applied by columns and the results are 3 FEC strings. They are used to fill the fields of the FEC packets as indicated in RFC 5109. As a practical example about the generation of the redundancy, let us consider the data vector resulting from the bits of the fields of "Padding Bit", that is the firs column of the strings in Figure 3.4.5. The information vector formed by padding bits is multiplied by the generator matrix G, and the results of multiplication is a code vector, where the 3 redundancy bits are the fields "Padding Bit" of the 3 FEC strings (or parity strings). This operation can be easily extended to every bit of the data strings, generating, in this way, the redundancy bits of each FEC string.

Although this type of architectures do not limit the use of any kind of code, the references are focused on linear systematic block codes. There are several standardized AL-FEC codes for FLUTE/ALC protocol [66], focused on file transfers, and FECFRAME [20], for streaming of real-time multimedia transfers. They are XOR-based scheme [25], Raptor/RaptorQ [40] [41], Reed-Solomon [30], and LDPC-staircase and triangle [50]. Consistently with these RFCs, the Chapters 4 and 5 contain a more exhaustive dissertation about how the recovery schemes, proposed in this thesis, protect the information data present in the header and in the payload of the RTP-media packets.

3.5 Transmission channel

It is important to remark that the nature of the IP packet losses in bursty. The causes can be summarized in two main categories, depending on the layers where the burst has been generated.

- At lower layers (physical for example): the most remarkable is the impulsive noise that is mainly produced by the switching in the electrical network. This can corrupt several milliseconds of the bit stream that, from the transmission point of view, means a burst of lost packets, since a corrupted packet corresponds to a lost packet. The length of the burst is related to the bit-rate of the communication.
- At higher layers. Due to the congestion of the net, the servers can discard several packets. Also in this case, the length of the burst is variable.

Hence, a very useful manner, for the purposes of this thesis, to characterize a communication channel can be based on two values: the probability of losing a packet, the Packet Error Rate (PER) and the average length of the bursts of lost packets (L_m). This two parameters are very important to design the FEC scheme, and, for this work, the considered transmission channel is based on a simplified Gilbert-Elliot model according to [67]. The two-state model that defines the simplified Gilbert-Elliot model is shown in Figure 3.5.1, where B indicates the bad state, i.e., the drop of a packet, and G represents the good state, a successful received packet. The transition probabilities are defined by:

$$\begin{aligned} p_{\mathrm{BG}} &= \frac{1}{L_m}; \\ p_{\mathrm{BB}} &= 1 - p_{\mathrm{BG}}; \\ p_{\mathrm{GB}} &= \frac{p_{\mathrm{BG}} \cdot \mathrm{PER}}{1 - \mathrm{PER}}; \\ p_{\mathrm{GG}} &= 1 - p_{\mathrm{GB}}; \end{aligned} \tag{3.5.1}$$

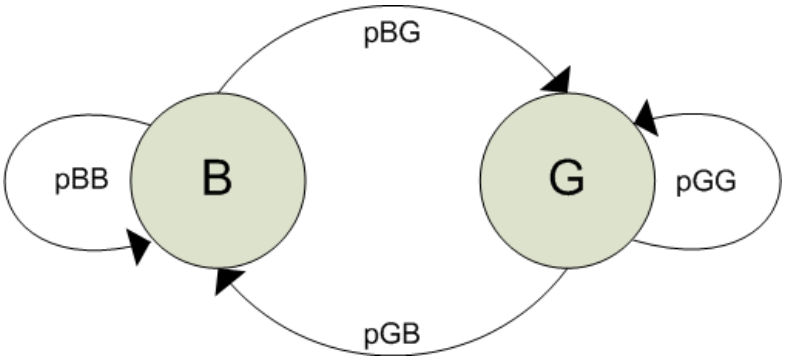


Figure 3.5.1: Simplified Gilbert-Elliot model.

More details about this aspect can be found in Section 4.4 and 5.4.1, where the use of the bursty-simulated channels are deeply discussed.

Chapter 4

Inter-Packet Symbol Approach to Reed-Solomon FEC Codes for RTP-Multimedia Stream Protection

In this chapter, an alternative Forward Error Correction scheme, based on Reed-Solomon codes, the inter-packet symbol approach, is presented. This scheme is based on an alternative bit structure that allocates each symbol of the Reed-Solomon code in several RTP-media packets. This characteristic permits to better exploit the recovery capability of Reed-Solomon codes against bursty packet losses.

The performance of this approach has been studied in terms of encoding/decoding time versus recovery capability, and compared with other proposed schemes in the literature. The theoretical analysis has shown that this approach allows the use of a lower size of the Galois Fields compared to other solutions. This lower size results in a decrease of the required encoding/decoding time while keeping a comparable recovery capability. Finally, experimental results have been carried out to assess the performance of this approach compared to other schemes in a simulated environment. Part of this work has been published as an article in [1].

4.1 Introduction

Reed-Solomon (RS) codes [22] [27], as already discussed more in depth in Section 2.3, are block codes that belong to the Maximum Distance Separable codes family. Block codes are defined by the parameters k, number of symbols of a data vector, and n, number of symbols of a code vector. Let $(d_0, d_1, ..., d_{k-1})$ be a data vector. A code vector is the result of the application of the code to a data vector. In case the code is systematic, the code vector is formed by appending the resulting $r = n - k$ redundancy (or parity) symbols to the data vector: $(d_0, d_1, ..., d_{k-1}, r_k, r_{k+1}, ..., r_{n-1})$ where r_i denotes parity symbols. Therefore, provided that the code vector has been correctly received, the use of systematic codes simplifies the decoding of the data vector, since just a direct extraction

from the code vector is required.

Reed-Solomon codes are linear no-binary cyclic codes, formed by sequences of m-bits symbols, that belong to 2^m extended Galois fields, where m takes values greater than 2. RS code parameters n and k are chosen so as to expression (2.3.10) is fulfilled:

- $n = 2^m - 1$;
- $k = 2^m - 1 - 2t$;

where $2t$ equals the redundancy r of the code.

The recovery capability of a MDS code depends on whether the positions of erroneous symbols are known (erasure codes), or not, (error detection and correction codes), as discussed in Section 2.2.3.1. In the first case, a maximum of r erroneous symbols can be recovered, whereas in the latter only $t = r/2$ symbols can be detected and corrected. From the point of view of the communication channel, this case is also known as PEC (Packet Erasure Channel), and it has been deeply argued in Section 3.1 of this thesis.

4.2 Protection scheme description

4.2.1 *Intra-packet symbol* approach

As already mentioned in Section 3.3, RTP (RFC 3550 [64]) is an application layer protocol that defines a packet format suitable for transmitting audio and video data over IP networks. It is commonly used on top of UDP and provides tools such as payload type identification, sequence numbering, or timestamping, which are useful for transmission management and monitoring. Nevertheless, RTP by itself does not provide any additional error protection mechanism.

In Section 3.4, it has been discussed how the protection schemes proposed in this thesis are situated in the FEC framework, contained in RFC 6363, in a general way, and how a FEC code has to be applied to follow the RFC 5109. Hence, in this Section more practical details about the architecture and the protection operation applied to the different RTP header fields and payloads for Reed-Solomon codes will be given. More specifically, for a given FEC code of parameters (n, k), the packetized multimedia stream (RTP-media packets) is divided into k-packet groups and the FEC code is applied to each group. The outcome of the FEC code is packetized in turn, in $n - k$ RTP packets (FEC-RTP packets) resulting in a new RTP stream that can be used at the reception side to recover lost RTP-media packets (see Section 3.4.1 and the example shown in Figure 3.4.5).

The fields of an RTP packet are combined to generate a bit sequence suitable for the application of the FEC code, as discussed in Sections 3.4.2 and 3.4.3, and as graphically summarized in Figure 4.2.1.

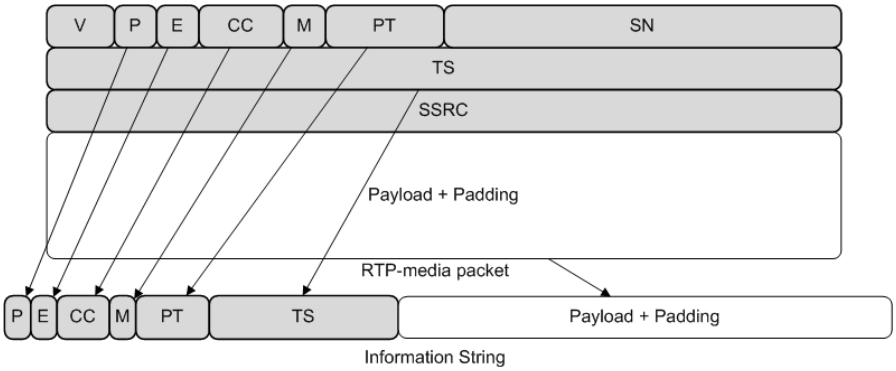


Figure 4.2.1: String generation from a RTP-media packet.

This output bit sequence is called *information string* (or *data string*). Therefore, from a group of k RTP-media packets, a set of k information strings are formed. Then, the selected FEC code is applied to them resulting in a new set of $n-k$ *parity strings* (or *FEC strings*). Finally, RFC 5109 specifies how to generate RTP-FEC packets out of the parity strings. Several FEC codes can be used according to this scheme. For instance, RFC 5109 includes examples of the application of simple XOR based codes.

In the literature, several works have attempted to apply Reed-Solomon codes to improve the robustness compared to other simpler codes. Authors in [68] consider a real-time Internet video context and applies an RS code orthogonally across k data packets (see Figure 4.2.2), producing $n - k$ redundant packets. In [69], a performance analysis of several FEC schemes, including RS codes, for real time applications is presented. All FEC proposed schemes generate $n - k$ redundant packets from each group of k data packets.

The use of RS codes requires to work with symbols of m bits. Therefore it is necessary to specify how the symbols are placed in the information strings. The protection scheme proposed in [68] [69] [70] suggests that each information string is divided into groups of m bits forming RS symbols (see Figure 4.2.2). In this thesis, this scheme is called *intra-packet symbol* approach. In this case, for a given RS code of parameters (n, k, m) ($RS(n, k, m)$, see Section 2.3.4 and definition 2.3.10), where m is usually 8, k RTP-media packets are needed to generate $n - k$ parity packets. Thus, this approach can recover up to $n - k$ lost packets out of each group of n transmitted packets (PEC, see Section 3.1). Note that RTP transmission results in an erasure code scenario since the location of packet losses is known. The recovery capability of the RS code is mainly controlled by the value of m according to 2.3.10. In the case of intra-packet symbol approach, the most common value of m reported in research literature is $m = 8$ (or $m = 16$) as a trade-off between computational complexity and recovery power.

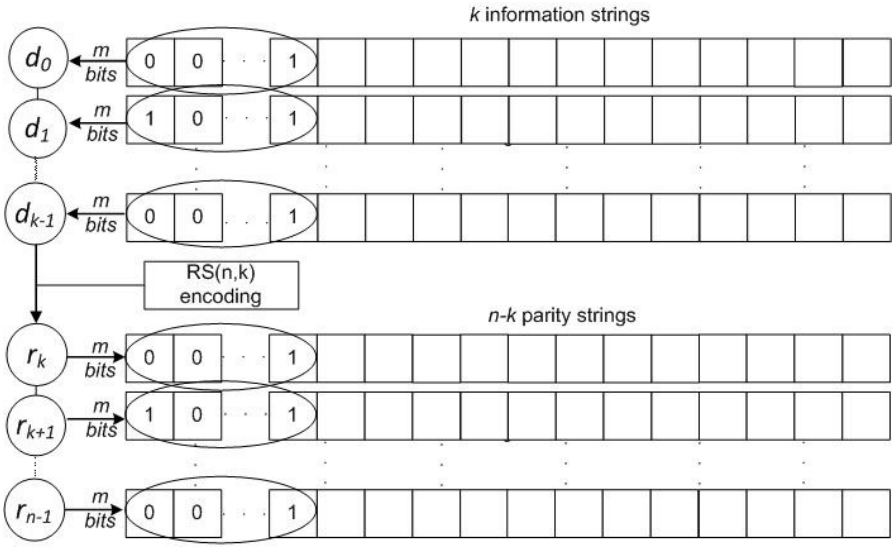


Figure 4.2.2: Symbols generation in intra-packets symbol approach.

Previous works, such as [29], have pointed out that RS codes suffer from computational complexity ([37] [69]) and the encoding/decoding time depends directly on the value of m, as it will be discussed in Section 4.3. For this reason, in this thesis, it is argued that an alternative approach to the application of RS codes to the information strings can be more convenient. The proposed approach allocates a given RS symbol along several information strings.

4.2.2 *Inter-packet symbol* approach

The main idea of the work illustrated in this Chapter is allocating RS symbols along several information strings as Figure 4.2.3 shows. For this purpose, this thesis proposes to work with groups of $k \cdot m$ RTP-media packets resulting in a matrix (information string matrix) of $k \cdot m$ information strings. Then it is considered that the m bits of each Reed-Solomon symbol are spread over different rows of the information string matrix. For this reason the scheme is called *inter-packet symbol* approach.

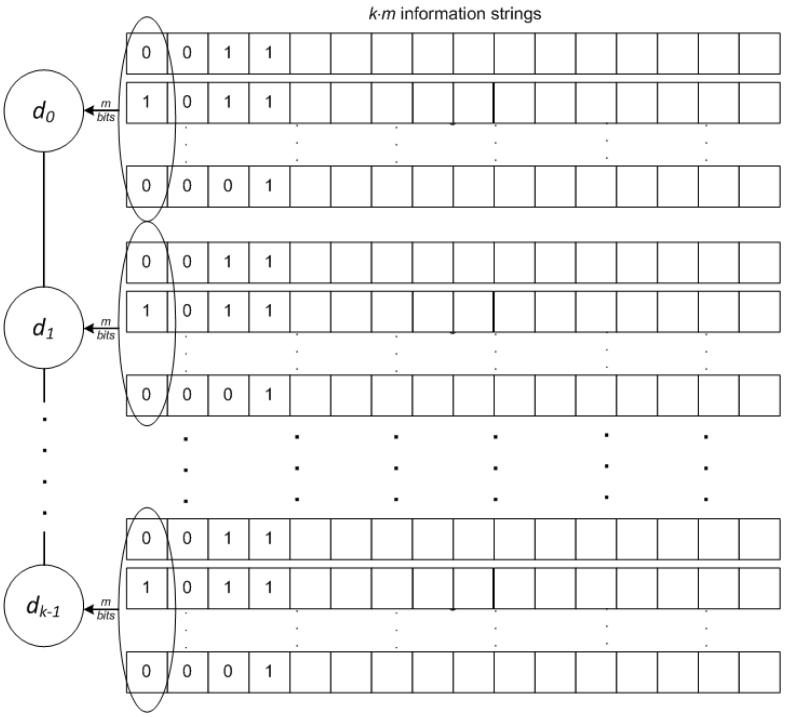


Figure 4.2.3: Symbols generation in inter-packet symbol approach.

Once the entire data vector is generated, $(d_0, ..., d_{k-1})$, it is encoded by the $RS(m, n, k)$ code as specified before, resulting in a code vector of n symbols. The first k symbols correspond to those of the data vector, and the remaining $n - k$ represents the redundancy coefficients as described in Figure 4.2.4.

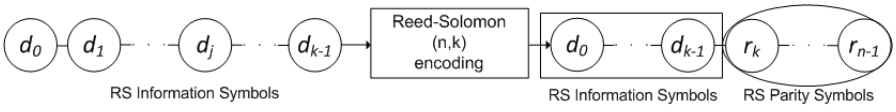


Figure 4.2.4: Encoding process for a data vector.

The redundancy coefficients $(r_k, ..., r_{n-1})$ are rearranged in a redundancy matrix of $(n-k) \cdot m$ rows as Figure 4.2.5 shows. Each row of this matrix represents a parity string, which is used to generate the corresponding FEC-RTP packet according to RFC 5109. Therefore, the outcome of protecting $k \cdot m$ RTP-media packets is a total of $(n-k) \cdot m$ FEC-RTP packets.

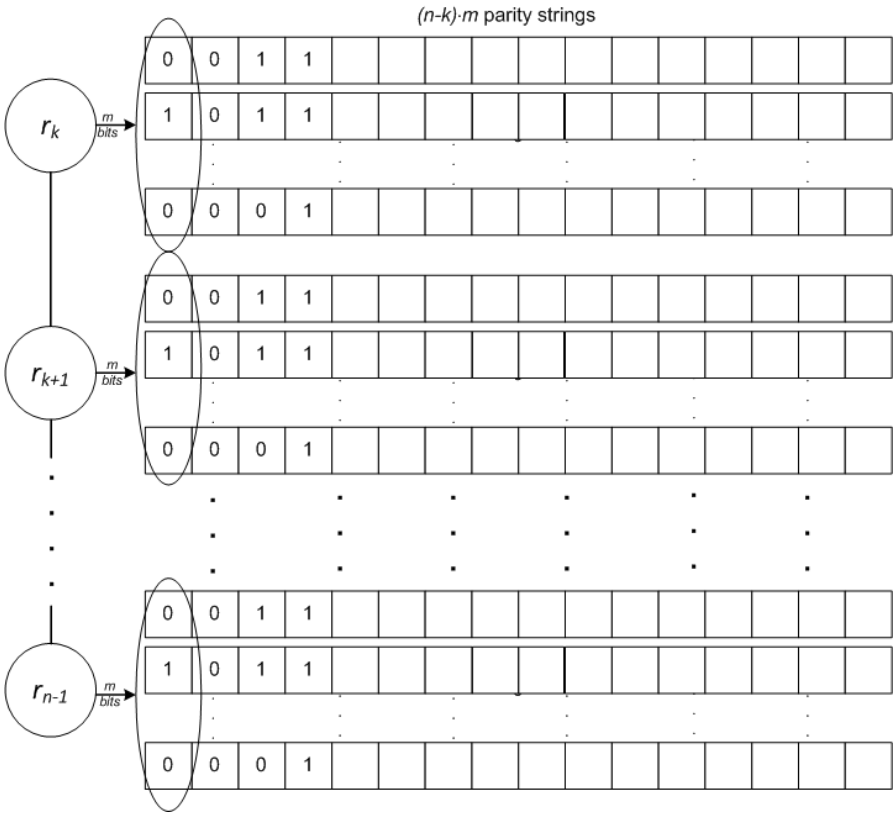


Figure 4.2.5: Generation of FEC-strings in inter-packet symbol approch.

This protection scheme permits to exploit better the recovery capability of Reed-Solomon codes against bursty packet losses, since a packet loss does not affect an entire symbol but only one of its bits. Indeed, the recovery capability of this approach can be up to $(n-k) \cdot m$, in the best case, and $(n-k) \cdot m - m + 1$, in the certain case. Therefore, the inter-packet symbol approach can reach similar recovery performance to that of the intra-packet symbol with a lower value of m, thus requiring less computational complexity. In the next Section the encoding time of both approaches is estimated and compared.

4.3 Performance analysis of the Reed-Solomon codes based schemes

In this Section a characterization of the computational complexity required by the intra-packet and inter-packet symbol approaches is presented. Based on the models on time devoted to encoding/decoding a data vector proposed in [29], the time required by both approaches to encode/decode a full media stream is characterized.

4.3.1 Encoding and decoding time of a single data vector

For a given systematic code of parameters (n, k), each data vector has to be used in the FEC code $n-k$ times in order to generate $n-k$ redundant symbols. Moreover, according to [29], the time to produce a single information symbol depends also on the size of the data vector, k. The bigger k is, the more time is needed to compute the parity symbols. Thus the encoding time of a data vector, t_e, can be expressed as a function of k and $n-k$, as (4.3.1) shows:

$$t_e = (n-k) \cdot k/C_e \tag{4.3.1}$$

where C_e is a constant that reflects the speed of the encoding system.

On the other hand, decoding time for a code vector depends on the number of missing transmitted symbols. So, considering the additional costs deriving from encoding matrix inversion, the decoding time, t_d, of a single symbol can be written as:

$$t_d = (n-k)/C_d \tag{4.3.2}$$

where the constant C_d is related to the speed of the decoding system. For a given FEC code, C_d can be considered smaller than its corresponding C_e, because of the decoding computational cost which includes the inversion of the encoding matrix.

As shown in (4.3.1) and (4.3.2), t_e and t_d have a very similar mathematical expression: they differ only by the multiplication values k/C_e and $1/C_d$. For a given FEC code, C_d can be considered smaller than its corresponding C_e, because of the decoding computational cost which includes the inversion of the encoding matrix: this means the decoding time for a single symbol is higher than the encoding time. Nevertheless, the total decoding time for an entire media sequence depends on the number and the position of the losses, so it is variable. On the contrary, the total encoding time has a general fixed expression, depending only on m and the length of the given media data content, two a priori known values. Therefore, since the equations (4.3.1) and (4.3.2) have a very similar behavior, in the following Section it will be only considered the expression of the encoding time t_e to study the performance of the proposed approach.

4.3.2 Encoding and decoding time of entire media sequence

Let us consider M as the total number of RTP-media packets needed to stream a given media content, and L, the number of bits of each information string.

In case of intra-packet symbol scheme, information strings are grouped in sets of k strings. In each set, the encoding operation on data vectors is applied L/m times, taking into account that each symbol in the data vector consists of m bits within each information string. Therefore, the number of times the encoding operations are applied for a whole content, N_o^{intra}, is expressed by the following equation:

$$N_o^{intra} = \frac{M}{k} \cdot \frac{L}{m} \tag{4.3.3}$$

Regarding the inter-packet symbol approach, RTP-media packets are organized in groups of $k \cdot m$ packets that generate $k \cdot m$ information strings. In this case, in each set of strings, the encoding operation on data vectors is applied L times, since the m bits of each symbol are spread over m different information strings (see Figure 4.2.3). Therefore, the number of times the encoding operations are applied for a whole content, N_o^{inter}, is expressed as:

$$N_o^{inter} = \frac{M}{km} \cdot L \tag{4.3.4}$$

Finally, the total encoding time for a media sequence from (4.3.1), (4.3.3), and (4.3.4) is computed:

$$t_e^{tot} = \frac{ML}{C_e} \cdot \frac{n-k}{m} \tag{4.3.5}$$

As can be seen in (4.3.5), the encoding time of an entire packetized media content follows the same expression for both schemes and it only depends on the parameters of the chosen RS code (n, k). In the case of the decoding time of an entire packetized media content that suffer losses, the reasoning is the same: the decoding time, for a fixed burst of erasure, depends on the values n and k, whereas the constant is related to the speed of the decoding system C_d, as shown in 4.3.2. Nevertheless, given a predetermined recovery capability that the protection scheme has to fulfill, the inter-packet approach is more efficient in terms of computational cost, given by lower values of m, the size of the associated Finite Field, than the classical approach. This can be achieved through the alternative distribution of the symbol bits along the information string, as it will be shown in this Section. Moreover, in order to not repeat the same analysis, as it has been argued in Section 4.3.1, the following discussion will be focused on encoding time.

Table 4.3.1 shows the different values given to the RS parameters that have been used to compare both approaches. Note that the recovery capability is conditioned by the number of resulting parity packets. Therefore those RS configurations that use comparable number of redundant packets and code rate have been selected. As a consequence, both schemes will require a similar number of input media packets to generate the FEC stream.

Table 4.3.1: Parameters m and n for intra-packet and inter-packet approach. m and n have been chosen so as to the number of redundant packets are comparable in both schemes.

Inter-Packet Approach	**Intra-Packet Approach**	**Parity Packets**
$m = 4$ bits per symbol $n \cdot m = 60$ total packets	$m = 6$ bits per symbol $n = 63$ total packets	$r_{pck} \in [6, 26]$
$m = 5$ bits per symbol $n \cdot m = 155$ total packets	$m = 7$ bits per symbol $n = 127$ total packets	$r_{pck} \in [5, 51]$
$m = 6$ bits per symbol $n \cdot m = 378$ total packets	$m = 8$ bits per symbol $n = 255$ total packets	$r_{pck} \in [7, 102]$

Figures 4.3.1, 4.3.2 and 4.3.3 show the total encoding time as a function of the number of generated redundancy packets (r_{pck}) in normalized time units:

$$Nor\, t_e^{tot} = (n - k)/m$$

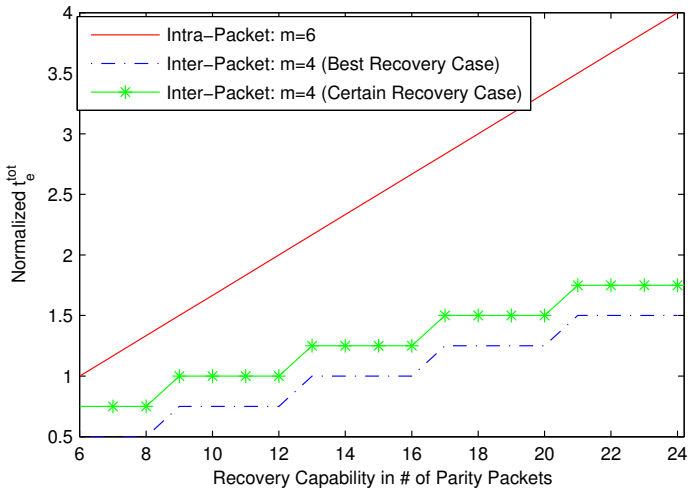


Figure 4.3.1: Encoding time of an entire video in two different schemes: Intra-Packet $m = 6$, Inter-Packet $m = 4$.

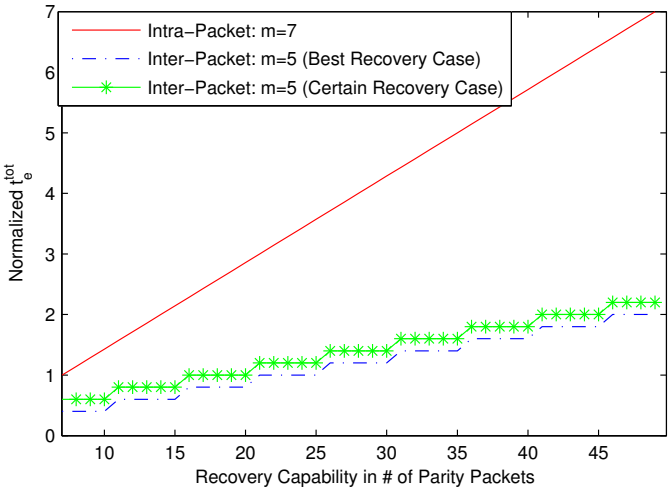


Figure 4.3.2: Encoding time of an entire video in two different schemes: Intra-Packet $m = 7$, Inter-Packet $m = 5$.

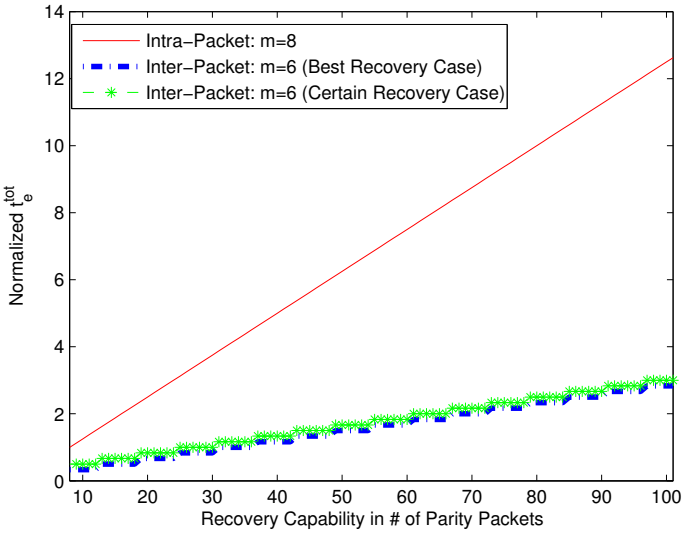


Figure 4.3.3: Encoding time of an entire video in two different schemes: Intra-Packet $m = 8$, Inter-Packet $m = 6$.

It can be observed that for a given recovery capability, the encoding time of the inter-packet symbol scheme is lower than that of the intra-packet symbol approach. This

occurs since the inter-packet symbol approach requires a lower value of m to fulfill a specific recovery capability. Note that the number of redundant packets in the proposed scheme is $r_{pck} = (n-k)\cdot m$, that provides a certain recovery capability of $(n-k)\cdot m-m+1$ packets, whereas in the intra-packet symbol is $r_{pck} = (n-k)$. In order to understand the variation of the recovery capability for the proposed scheme, an example for a code vector, with parameters $m = 3$, $n = 7$, $k = 4$, is proposed. The recovery capability is $n - k = 3$ in the case of erasure. Depending on how the burst of lost packets starts and its length, the recovery capability, as number of bits (and as a consequence number of packets for the proposed scheme), changes. If a burst of losses that starts just at the beginning of a symbol and its length is $(n - k) \cdot m = 9$ bits is considered, the number of symbols that can be recovered is $n - k = 3$, as shown in Figure 4.3.4.

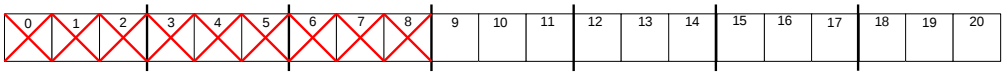


Figure 4.3.4: Burst of erasures, length $(n - k) \cdot m$, starting at the beginning of a symbol.

Whereas, if the same burst of erasures starts at the second bit, the number of damaged symbols is $n - k + 1 = 4$, hence, the code is not able to recover all the losses, as shown in Figure 4.3.5.

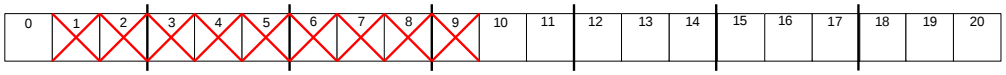


Figure 4.3.5: Burst of erasures, length $(n - k) \cdot m$, starting inside a symbol.

4.4 Simulation and results

The recovery capability of the inter-packet symbol approach depends on how bursts affect each group of packets since each packet contains a single bit of a symbol. Therefore, a lower bound can be computed being the maximum length of a burst of lost packets that the inter-packet symbol approach can certainly recover (the certain case). Nevertheless, in case the burst is aligned with a packet that corresponds to the beginning of a symbol (the best case), the maximum length of the burst that can be recovered is higher than that of the certain case. Therefore, an RS code following the proposed approach can recover error bursts of length ranging from $((n - k) \cdot m - m + 1)$ packets in the certain case up to $((n - k) \cdot m)$ packets in the best case.

In order to assess the effectiveness of the proposed approach within a communication system, two RS codes in a simulated environment have been compared. The m parameters of both protection schemes are $m = 4$ in case of inter-packet symbol approach and

$m = 6$ in case of intra-packet symbol scheme (see Table 4.3.1). Besides, the code rate has been fixed to $k/n = 40\%$ for both approaches. The recovery characteristics of both RS codes are summarized in Table 4.4.1.

Table 4.4.1: Recovery capability for Intra-Packet and inter-packet approach.

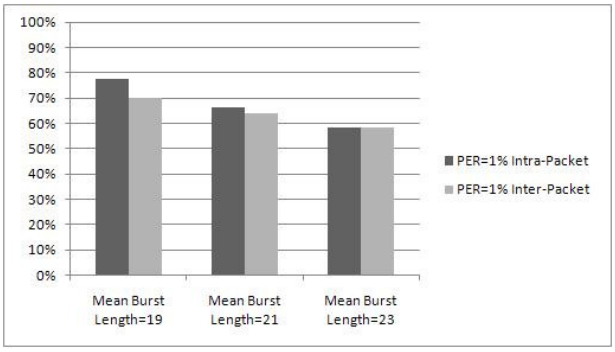
Intra-Packet Approach	**Inter-Packet Approach**	
24 packets	***best case***	***certain case***
	24 packets	21 packets

Different experiments simulating the transmission of an RTP-media stream together with its corresponding RTP-FEC stream have been carried out. The input to system used in this thesis is an MPEG2-TS video movie. The transmission channel is simulated through a simplified Gilbert-Elliot model according to [67], as indicated in Section 3.5. The parameters that defines the model are the probability of losing a packet, the Packet Error Rate (PER), and the average length of the bursts of lost packets (L_m). Table 4.4.2 shows the parameters of the channel models considered. The average burst length has been set close to the maximum recovery capability of the RS codes and close to the typical average burst length for wireless networks (about 20 packets for 802.11g [9]) .

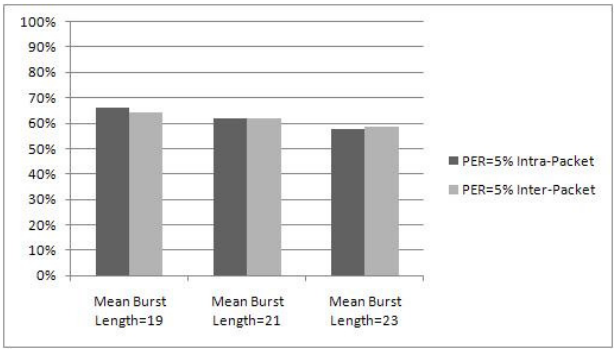
Table 4.4.2: Parameters of communication channel.

Packet Error Rate (PER)	**Average Burst Length (packets)**
1%	19 21 23
5%	19 21 23
10%	19 21 23

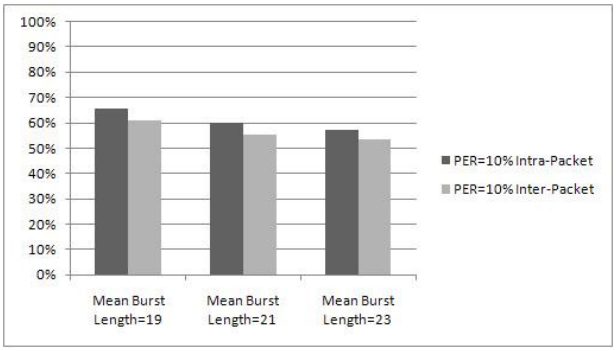
Figure 4.4.1 shows the ratio of recovered packets with respect to the number of lost packets. As can be observed, the inter-packet symbol approach provides similar results in all the experiments to those of the intra-packet symbol scheme. Moreover, the results of the inter-packet approach are closer to those of the intra-packet approach as the average burst length increases, although the value of m is lower.



(a) PER=1%



(b) PER=5%



(c) PER=10%

Figure 4.4.1: Percentage of recovered packets obtained by the inter-packet and intra-packet approaches for the different channel models specified in Table 4.4.2.

4.5 Conclusions

In this a novel approach to handle Reed-Solomon codes within a protection scheme intended for RTP transmission of multimedia contents has been introduced: the inter-packet symbol approach. This scheme is based on an alternative bit structure that allocates each symbol of the RS code along several RTP-media packets. This characteristic permits to exploit better the recovery capability of Reed-Solomon codes against bursty packet losses, since a packet loss does not affect an entire symbol but only one of its bits. Moreover, the inter-packet symbol approach is compatible with the RFC 5109.

The performance of the proposed approach have been analyzed in terms of computational complexity versus recovery capability, and compared with other proposed schemes in the literature that follow an intra-packet symbol approach. The theoretical analysis in Section 4.3.2 has shown that the proposed approach allows the use of a smaller size of the Galois Fields size compared to other solutions. This lower size results in a decrease of the required computational cost while keeping a comparable recovery capability. This result has been finally assessed through experimental tests in which both schemes have been used to protect an RTP-media transmission in a simulated wireless environment.

Chapter 5

Low Latency LDPC Code for Multimedia-Packet Stream in Bursty Packet Loss Networks

5.1 Introduction

LDPC codes can recover multiple losses and they are suited for channels where losses occur following a uniform distribution of probability, and each loss is independent from all the other ones, i.e., memoryless channels [71]. Comparing LDPC codes with Reed-Solomon ones, although LDPC codes are sub-optimal in terms of recovery capabilities [37], they require a lower computational cost. Indeed, they need linear or quasi-linear encoding/decoding complexity [36], instead of Reed-Solomon codes, that have a high complexity, since the use of Galois Fields [29].

Moreover, LDPC codes are defined as large block codes [52], since their robustness increases with the number of information packets involved. Nevertheless, using a large number of information packets implies an increase the latency, which may be inconvenient for real-time applications.

Hence, the main goal of this Chapter is optimizing this type of codes to channels with memory, that is, where losses occur in bursts (IP networks). In addition, codes that involve a small number of information packets in an attempt to reduce the required latency are considered.

Another complementary objective of this Chapter is to carry out an analysis that allows the evaluation of the recovery capability of an LPDC code under these conditions (bursty channels). Since the structure of this family of FEC codes is defined by a randomly generated parity matrix that identifies which packets are involved in the generation of each FEC packet, this evaluation is essential to achieve the main objective of this Chapter, that is, the optimization of these codes for bursty erasure channels. As this is a stochastic process, these matrices might not be uniformly robust, hence an analysis that is capable of assessing those parts of the parity matrix that are weaker against bursty losses and those parts that are stronger, is needed.

Part of this work has been published as articles in [2] [3] [4] [5] [6].

5.2 State of the art for LDPC codes in bursty channels

In the literature, several approaches that use this family of codes in scenarios with channels with memory can be found. In this sense, the authors in [72] propose a technique that allows to construct an LDPC matrix intended for burst erasure correction, using a superposition scheme that consists in replacing the entries of a base matrix H_{base} with binary matrices called superposition matrices [73]. Each 0 entry is replaced by a 0's superposition matrix, while each 1 entry is replaced by a circulant or permutation matrix. The recovery capability depends on the characteristics of the selected permutation matrices. In this thesis a simpler alternative is proposed in order to create an H matrix using classical pseudo-random algorithms, easy to find in the literature, and afterwards modify H in order to make it more robust against bursts of lost packets. This is done by means of a novel characterization of its recovery capability in bursty channels. Another approach for creating the parity matrix is presented in [74]. They propose two algebraic methods for the systematic construction of quasi-cyclic LDPC codes. However, they evaluate their recovery performances for very large values of blocks, while in the approach of this thesis, a significantly smaller value is considered. In other works [75] [76], it is also proposed to modify the parity matrix using a column permutation algorithm. The authors propose different metrics to estimate the recovery capabilities of the parity matrix. Again, these works consider a larger number of information packets than that of the approach of this thesis. On the other hand, the authors in [50] [52] [58] propose to randomize the order of the transmission of the packets of the stream. This technique allows to decorrelate the packets affected by bursts and reaches good results for a high number of blocks. On the contrary, for low values of this parameter, the results of this chapter show that the modified matrices perform better in bursty channels. Other works address the application of LDPC to bursty channels, focusing on alternative more complex decoding algorithms, like those in [77] [78], or encoding algorithms, like the one in [79], as a mean to recover from error bursts.

5.3 Small block LDPC codes in low-latency memory channels

5.3.1 Protection scheme

As Section 3.3 explains in depth, in a typical video transmission model based on IP/UDP/RTP protocols, the encoded video stream is encapsulated in RTP media packets that, in turn, are encapsulated in UDP.

The protection system works at the application layer, and it follows the architecture presented in the FEC framework [20], as mentioned in Section 3.4.1. This means that two RTP instances are generated: one for the source packets and another one for the repair packets (RTP-FEC packets). Thus, an RTP-FEC packet contains an RTP header

of its own, and the redundancy data for the RTP source header and its payload. The information protecting header and protecting payload media data is generated by applying the FEC code across the RTP source packets. In Section 3.4.2 of this thesis there are the practical details of this operations for a general case, whereas, in Section 4.2.1, there is a particularization of these operations to a Reed-Solomon-based approach.

In this work, each block of k data packets is encoded using an LDPC code. The selected LDPC codes are the so-called low-density generator matrix (LDGM) codes, which are a simplified version of LDPC codes. They are more deeply analyzed in Section 2.4.5.2 of this thesis. It is important to remind that the main characteristic of this type of LDPC codes is that matrix G and matrix H coincide (see Section 2.4.5.2). Moreover, in [52], the use of this type of codes is addressed to a very large value of k, whereas in this work, it is significantly lower. The utilization of LDGM codes is very useful from a didactic point of view. Thanks to their high simplicity, it is trivial to understand how the modifying algorithm works for low values of k, since a modification of the H matrix has a direct impact on the G matrix. As discussed in Section 2.4.5, in the generic LDPC codes, the creation of the G matrix from the H matrix is not trivial and any modifications in H are not reflected in G. In the LDGM codes, H is an $(n-k) \times k$ matrix, whereas in the case of classical LDPC codes, it is $(n-k) \times n$ (see Section 2.4.5.2). The bipartite graph of LDGM codes is shown in Figure 5.3.1.

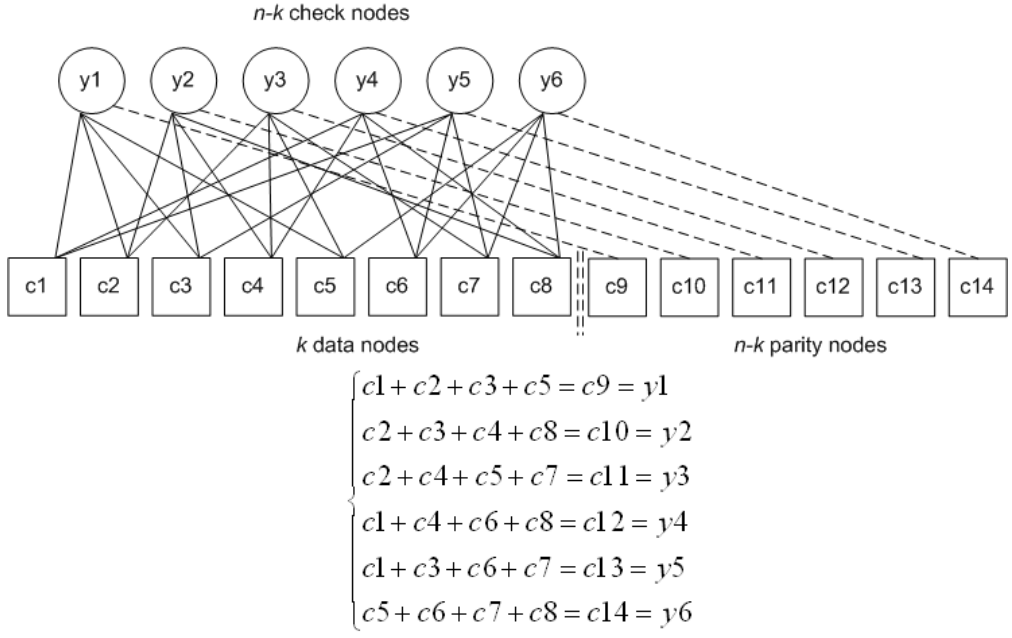


Figure 5.3.1: LDGM code. Example of bipartite graph for an LDGM code.

A regular LDGM code has a fixed number of 1's per column and a fixed number of 1's per row. The equation (2.4.9), $k \cdot w_c = (n-k) \cdot w_r$, expresses this condition. In this

expression, the edges that link parity nodes to check nodes are not considered in w_c and w_r.

In LDGM codes, a recovery packet is generated for each row of H, as the result of applying the XOR operation to the data packets corresponding to the entries equal to 1 in H (Figure 5.3.2).

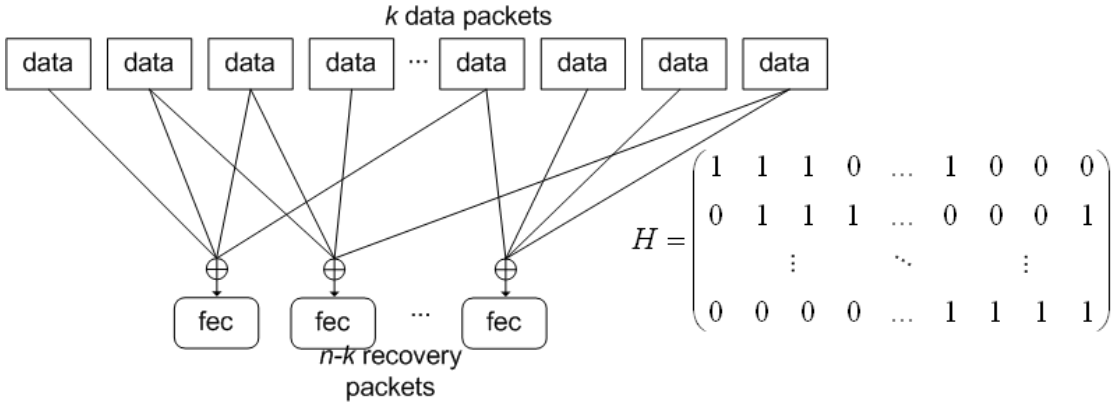


Figure 5.3.2: FEC generation. Protection operations for a block of k data packets.

The three main steps of this protection scheme are illustrated in Figure 5.3.3: (i) division of data packets into blocks of low k, (ii) encoding of each block of k data packets, and (iii) the outcome of the encoding for each block of k data packets forms a set of $n-k$ packets.

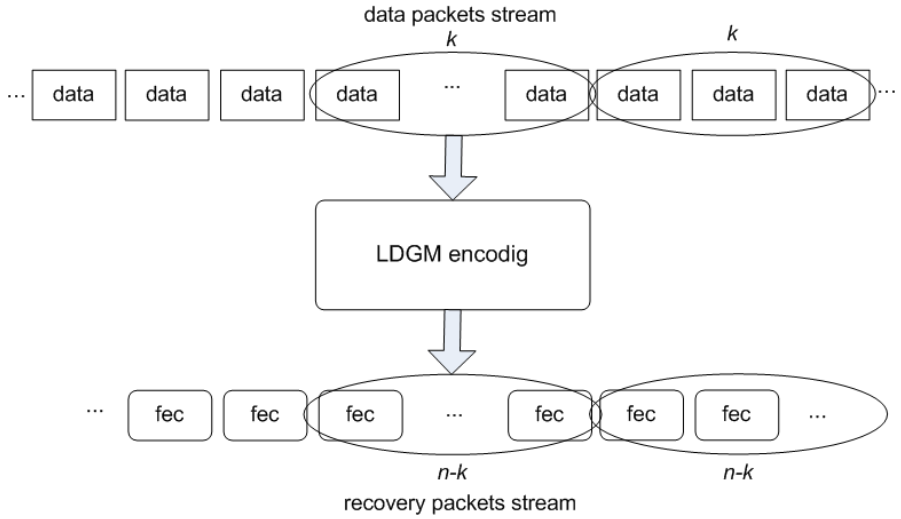


Figure 5.3.3: Protection scheme. Example of the packetized protection scheme based on LDGM codes.

5.3.2 Low-Density Burst-Oriented Generator Matrix

As said before, it is not possible to exactly compute the a priori recovery capability of this family of codes from the value of the parameters k and n. For this reason, one of the most essential steps of this work is the design of an algorithm that evaluates the global recovery capability of a randomly generated LDGM code for a bursty channel.

This algorithm firstly estimates the contribution of each column of the matrix H to the global recovery capability of the LDGM code (see Section 5.3.2.1). With that measurement, it is possible to identify those parts of the parity matrix that are more sensitive to bursty losses and those parts that are stronger. Moreover, aggregating the values of all the columns of matrix H, a global measurement of the recovery capability of the matrix is obtained.

Finally, taking into account sensitive parts and strong parts, in a second step, the proposed algorithm modifies them with the aim of improving the general recovery capability of the LDGM code keeping its intrinsic parameters, k, n, w_c, and w_r.

5.3.2.1 Column-based Recovery Measurement for matrix H

The use of a simplified model to estimate this measurement is proposed. The main assumption is that only a single burst can occur per block of k packets. Note that, for low values of packet error rate (PER), this assumption is reasonable for small values of k, as shown in [65].

Let us consider a block of k packets to be protected following the rules indicated by matrix H (Figure 5.3.4).

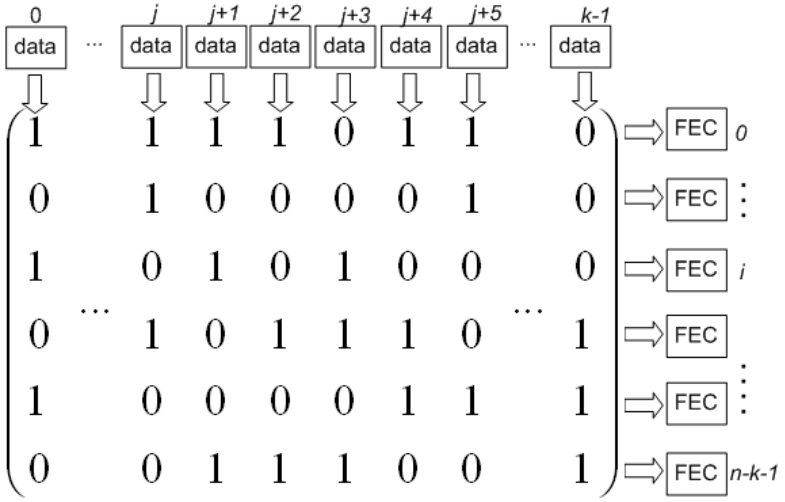


Figure 5.3.4: Correspondence column packets.

For each column j, all possible bursts that start at the packet in position j and whose lengths range from 2 packets to $n-k$ packets are considered. For each burst, whether the lost packets can be recovered or not are considered. The recovery capability of column j (Column-based Recovery Measurement, CRM) as the number of recovered bursts (the higher this value is, the more recovery capability column j has) is estimated. Formally:

$$\mathrm{CRM}(j) = \sum_{l=2}^{\max(n-k,n-j)} \mathrm{FEC}(B_j^l); \tag{5.3.1}$$

where B_j^l is a burst of length l, $l \in (2, n-k)$, starting in packet j, $j \in (0, k-1)$, and $\mathrm{FEC}(\cdot)$ indicates whether the lost packets have been recovered by the LDGM code ($\mathrm{FEC}(\cdot) = 1$) or not ($\mathrm{FEC}(\cdot) = 0$).

Hence, the algorithm is able to characterize the recovery capability of H in a column-based manner. Those columns that have a low value of CRM are tagged as the weakest parts of H matrix , whereas the strongest parts are represented by the columns that achieve a high value of CRM.

Figure 5.3.5 shows an example of $\mathrm{CRM}(j)$. In this example, it is easy to see the parts where local minima and local maxima of CRM are located.

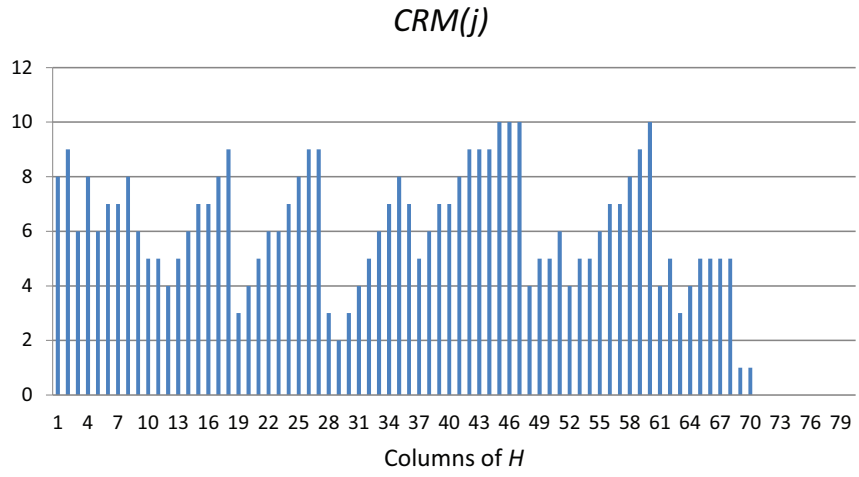


Figure 5.3.5: $\mathrm{CRM}(j)$. Number of recovered bursts of lost packets for each position inside a block of k packets, which correspond to the columns of H, that is $\mathrm{CRM}(j)$.

5.3.2.2 Global Recovery Measurement

A global measurement of recovery capability is given: the total amount of bursts that can be recovered is computed for a given matrix H by the Global Recovery Measurement

(GRM), defined as:

$$\mathrm{GRM}(H) = \sum_{j=0}^{k-1} \mathrm{CRM}(j) \tag{5.3.2}$$

Burst-oriented modifying algorithm

Once the weakest and the strongest parts of the matrix H are identified, the following step of the algorithm consists in identifying the global minimum of CRM(j) of H ($\mathrm{CRM}_{\min}(j)$), that is a minimum of recovery capability in the matrix.

The objective is to increase the value of $\mathrm{CRM}_{\min}(j)$ with the aim of improving the global recovery capability of the matrix, that is, increasing the value of GRM(H). This operation is performed by a local analysis around column j. For that purpose, a burst of lost packets that begins in j and has an arbitrary length l is considered in order to test how it affects the recovery capability of matrix H. This means that the algorithm considers a sub-matrix within H, that starts in column j and terminates in column $(j + l - 1)$, and checks which rows have more than one packet affected by the burst. Those rows are useless to recover lost packets, since each row is able to recover just one lost packet. Therefore, the algorithm aims at modifying the equations defined by H in order to be able to recover the packets affected by the artificial burst.

Let us consider the example in Figure 5.3.6, where the case of a four lost-packet burst (packet j to packet $j + 3$ are lost) is shown. The LDGM code employed in the example cannot recover this burst. However, although Eq. 4 is able to recover packet $j+3$, which can be employed in Eqs. 0 and 3, Eqs. 0, 2, 3, and 5 have more than one lost packet, so they are useless to recover packets j, $j + 1$, and $j + 3$. If the right packet in those equations whose packets are not involved in the burst is involved, the code would be able to recover more lost packets using the remaining equations.

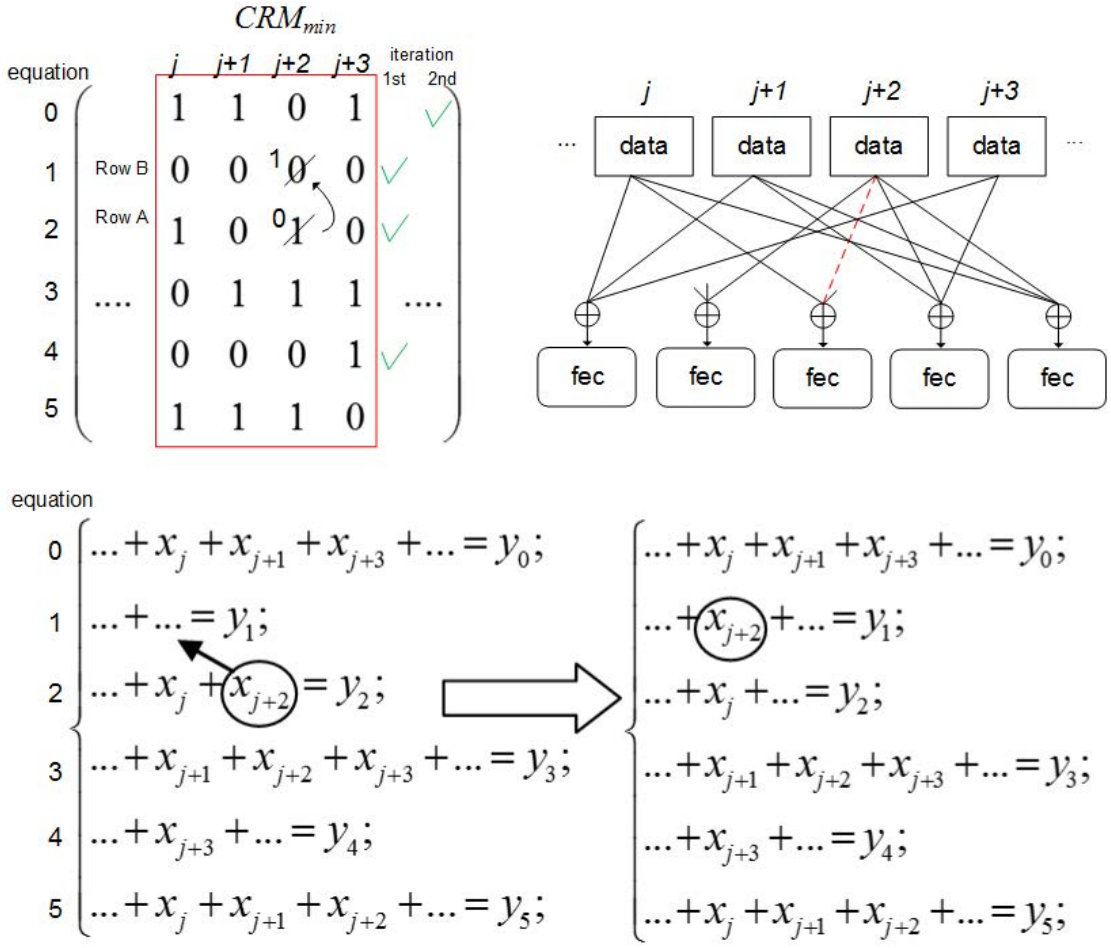


Figure 5.3.6: H modification. Example of how to modify an H matrix in order to achieve a best recovery capability.

In detail, the algorithm moves the second 1 entry of Eq. 2 to Eq. 1 (generically, from row A to row B), in order to employ the packet $j+2$ in the generation of the second FEC packet, as shown in Figure 5.3.6. In this new case, the updated LDGM code is able to solve Eqs. 1, 2, and 4 during the first iteration of the decoding algorithm. This means that packets $j+2$, j, and $j+3$ can be recovered, since there is only a lost packet per row. In the second iteration, it is possible to solve Eq. 0 (packets j and $j+3$ are already known) recovering packet $j+1$. The algorithm operates this type of changes in the sub-matrix of H that starts in column j, where CRM_{min} is situated, until column $j+l$, with l defined by the characteristics of the bursty channel.

Finally, in order to keep constant the value of w_r for every row of H, the algorithm has to move a 1 entry from row B to row A, in a different part of the matrix, as shown

in Figure 5.3.7, which refers to the example presented in Figure 5.3.6. Since this change should not penalize the recovery capability of this part of the matrix, the best option to move the 1 entry from is the sub-matrix of H that starts at column h, where CRM_{max} is, to $h+l$. Typically, this part is more resilient against modifications than other parts, considering that a larger value of CRM entails a higher robustness against bursts. That means that packets are better organized within the equation.

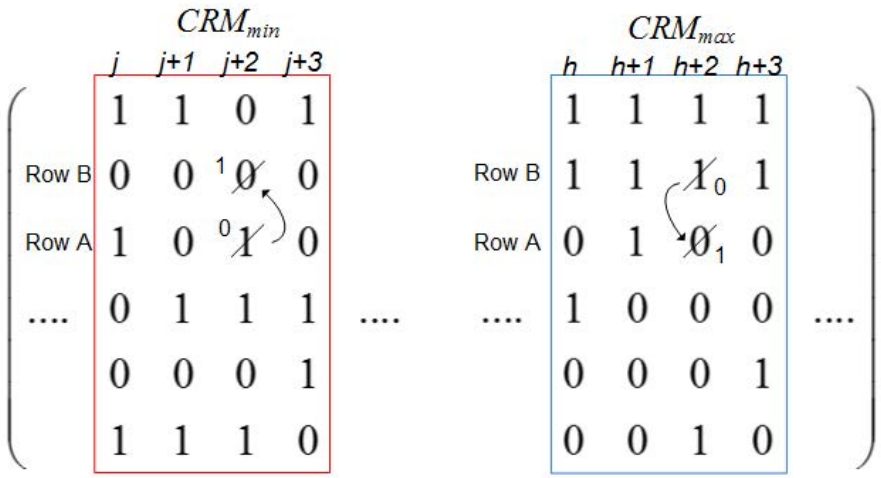


Figure 5.3.7: Fixed number of 1 entries per row. Example of how to keep the fixed number of 1 entries per row (w_r) without modifying the recovery capability of H.

This operation (modifying the part of the matrix where CRM_{min} is placed, in order to improve its value) changes $GRM(H)$. The algorithm iterates while $GRM(H)$ keeps on increasing: when this global parameter decreases, the algorithm stops modifying the H matrix. The low-density burst-oriented generator matrix (LDBOGM) is the H matrix that achieves the best $GRM(H)$ value that the algorithm is able to reach. Formally, let $GRM_t(H)$ be $GRM(H)$ before the iteration t, and $GRM_{t+1}(H)$ its updated value. Then, the algorithm is iterated till:

$$\mathrm{GRM}_{t+1}(H) < \mathrm{GRM}_t(H) \tag{5.3.3}$$

The main steps of the algorithm are outlined in Table 5.3.1, and Figure 5.3.8 shows all the process of the burst-oriented modifying algorithm by a block diagram.

1. Find the global minimum of CRM(j), $\text{CRM}_{\min}(j)$;
2. Define a window within H that has as first column the corresponding column of $\text{CRM}_{\min}(j)$, the same number of rows of H, and a predefined number of columns l (a window of columns);
3. Solve the equations system H without the packets (variables) used in the sub-matrix of the global minimum;
4. Point out which equations (rows) cannot be solved;
5. Move the 1 entry belonging to an equation unsolved (row A) in the sub-matrix of the global minimum to another equation (row B). Row B is a row that does not loose its recovery capability with this change. The objective is to solve the equations system (as in the example in Figure 5.3.6 where row A is the third one, and row B is the second one);
6. Find the global maximum of CRM(j), $\text{CRM}_{\max}(i)$;
7. Define a new sub-matrix that starts in the column of the global maximum and has the same dimensions of the previous matrix;
8. Move 1 entry from row B to row A, in order to preserve w_r for all the columns of H;
9. Is $\text{GRM}_{t+1}(H) < \text{GRM}_t(H)$?

Table 5.3.1: Main steps of burst-oriented modifying algorithm.

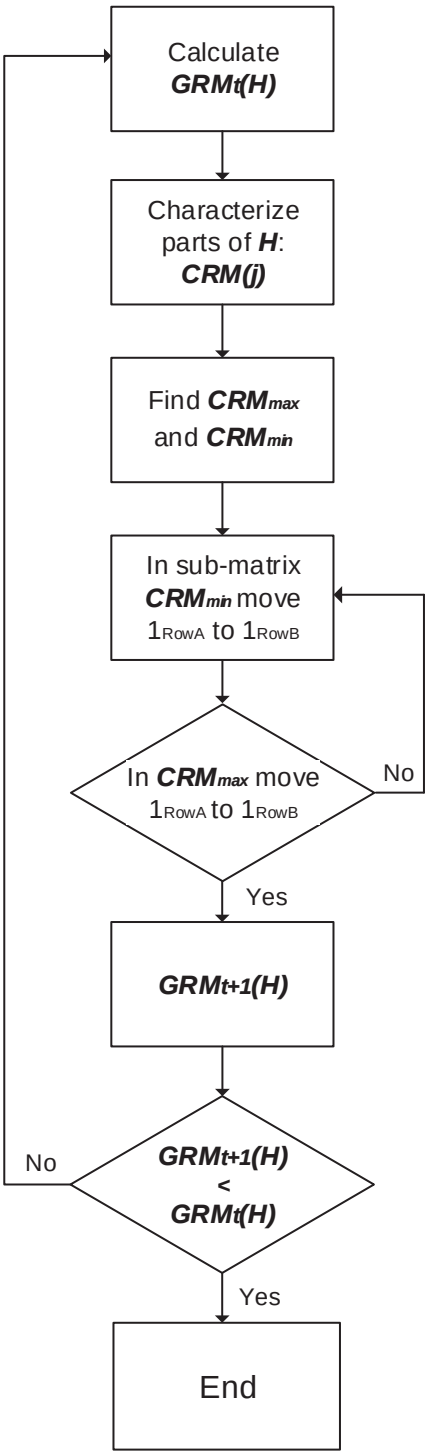


Figure 5.3.8: Burst-oriented modifying algorithm. The block diagram that outlines the main steps of burst-oriented modifying algorithm.

Finally, it is meaningful to underline that the decoding operations are based on a simple iterative algorithm: given a set of linear equations, if one of them has only one unknown variable, then its value is that of the constant term. This variable is replaced in all remaining linear equations, and the algorithm reiterates. Hence, several unknown variables can be found by the recursive algorithm. The modifying algorithm does not modify the original iterative decoding algorithm, since the modified generator matrix is known both at encoder and decoder.

5.4 Simulations and results

5.4.1 Design parameters

A crucial aspect in the configuration of any channel code is the appropriate selection of parameters k and n according to the characteristics of the communications channel used. In the approach of this work, the choice of these parameters is determined, on the one hand, by the statistical characteristics of the channel, and, on the other hand, by the imposed minimum code rate. The aim of this choice is to guarantee the best achievable recovery capability with the lower added redundancy.

The selection of k is particularly relevant because its value directly affects the latency generated at the receiver, becoming a problem in the case of time-sensitive services, such as videoconference. Thus, it is very useful to employ low values for this parameter. In that sense, the approach in [65] employs an interleaving protection scheme with $k = 80$ and is able to recover bursts of up to 20 packets. These values depend also on the rate of the provided services. A typical value proposed for bitrates of about 10 Mbps and for XOR-based interleaved systems is about $k = 100$ [25].

Regarding n, this parameter determines the amount of added redundancy and the recovery capability. Typical reasonable values are those that achieve a redundancy rate about 15 and 20% of the total information [65]. Moreover, it is important to underline that this work is focused on very low values of k (and n). Whereas in literature, very high values of k are mostly proposed, in this scheme $n = 100$ and $k = 80$.

The considered transmission channel is based on a simplified Gilbert-Elliot model according to [67], and as indicated in Section 3.5: the main statistical characteristics are represented by the PER (Packet Error Rate) and the average burst length (L_m) in number of packets.

5.4.2 Simulations

5.4.2.1 Analysis of the recovery capability of LDBOGM codes

In order to provide a comprehensive evaluation of LDBOGM matrices, the analysis firstly needs the generation of 50 different matrices using the classical stochastic algorithm in [80] and in [81], keeping n, k, and w_c fixed to values determined in Section 5.4.1. Every

LDGM matrix has been modified using the proposed algorithm, and a corresponding LDBOGM matrix has been generated. Afterwards, for each pair of matrices (original and modified), the recovery capability in a set of experiments has been evaluated. This has been done it through the ratio of recovered packets with respect to lost packets.

The experiments have consisted in simulating the transmission of an RTP media stream together with its corresponding FEC stream. The recovery capability is calculated after a total amount of 2000 blocks of k transmitted packets, in order to achieve statistically relevant results.

The experiments have been carried out for four different channels as representative of the conditions of wired and wireless networks not only in a normal context but also in congested channels [9] [65] [82] [83]. In particular, the channels have been considered with two different average burst lengths, $L_m = 5$ and $L_m = 10$, and two values of PER, 1 and 5%.

Figure 5.4.1 shows the obtained results for the LDGM and LDBOGM schemes.

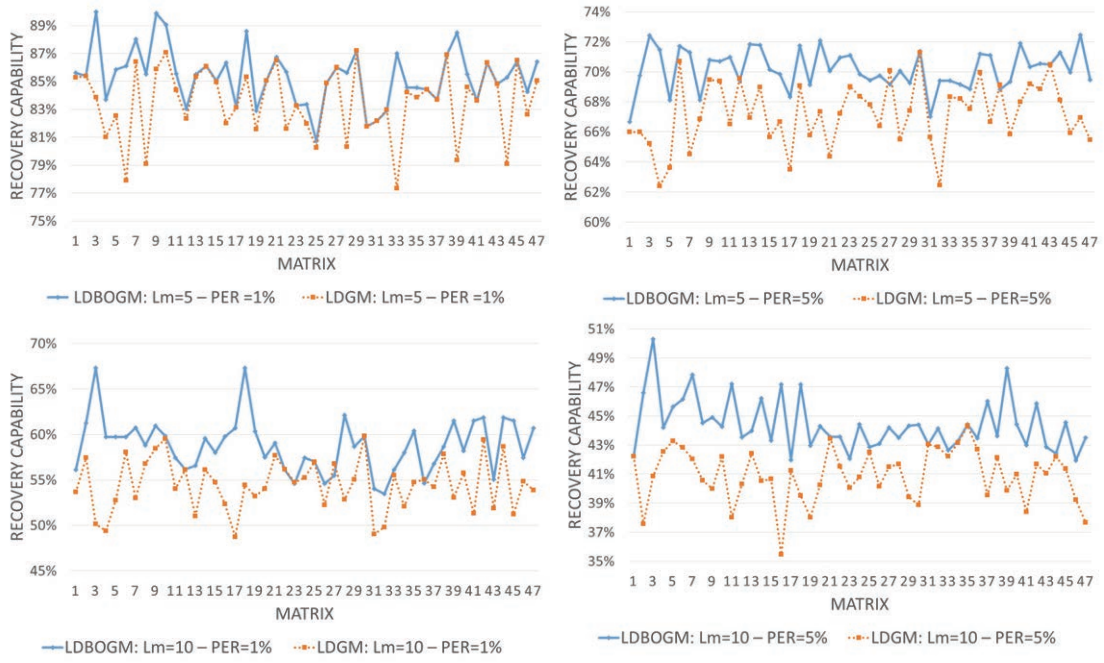


Figure 5.4.1: LDBOGM vs. LDGM. Percentage of recovered packets for 50 different matrices for four different channels.

As can be observed, LDBOGMs obtain better results in almost all the experiments. In the best case, the proposed algorithm is able to improve the recovery capability of an LDGM code by up to 10%. In the worst cases, the recovery capability remains the same.

In Tables 5.4.1 and 5.4.2, the best, the worst, and the average recovery capability for both schemes are indicated.

Table 5.4.1: Recovery capability of LDBOGM.

	$L_m = 5$ PER=1 %	$L_m = 5$ PER=5 %	$L_m = 10$ PER=1 %	$L_m = 10$ PER=5 %
Max.	90 %	72.5 %	67 %	50 %
Min.	81 %	67 %	52 %	42 %
Average	85 %	70 %	58 %	44 %

Table 5.4.2: Recovery capability of original LDGM

	$L_m = 5$ PER=1 %	$L_m = 5$ PER=5 %	$L_m = 10$ PER=1 %	$L_m = 10$ PER=5 %
Max.	87 %	71 %	63 %	45 %
Min.	77 %	62 %	49 %	35 %
Average	83 %	67 %	55 %	41 %

Moreover, in the considered scenario, the modification of the matrix is done off-line. Nevertheless, it is interesting to outline that, given $k = 80$, $n = 100$, and $w_c = 3$ (values used in the simulations), the refinement time, calculated as the average time for 50 matrices, is 69.1476 ms. Whereas, the original H matrix creation time is 0.0615 ms. The time measurements are implemented in C++, and they have been carried out in a PC with an Intel Core i7-3540 @ 3 GHz.

5.4.2.2 Comparative results

To assess the effectiveness of the proposed approach, the selected modified matrix is the case that has obtained the best average recovery capabilities in the analysis presented in the previous Section (LBOGM) and compared it with: (i) the original matrix, (ii) an XOR-based 2D interleaved code, with 10 columns and 8 rows (identified by Interleaving), and, finally, (iii) a Reed-Solomon scheme, based on the inter-packet approach with $m = 4$ and $n = 60$ proposed in [1], as deeply discussed in Chapter 4 (identified by Reed-Solomon).

The experiments have been carried out following the same methodology described in the previous Section, but the evaluation has been extended to a much larger set of channel models. In this sense, a PER ranging from 0.1 to 20% has been considered, and the average burst length has been extended up to $L_m = 20$. The reason is that in the estimations used for this thesis, burst lengths from 2 to 20 packets are considered. The actual used values are the following: $L_m = 5$, 10, 15, and 20 and PER =0.1, 1, 5, 10, 15, and 20%.

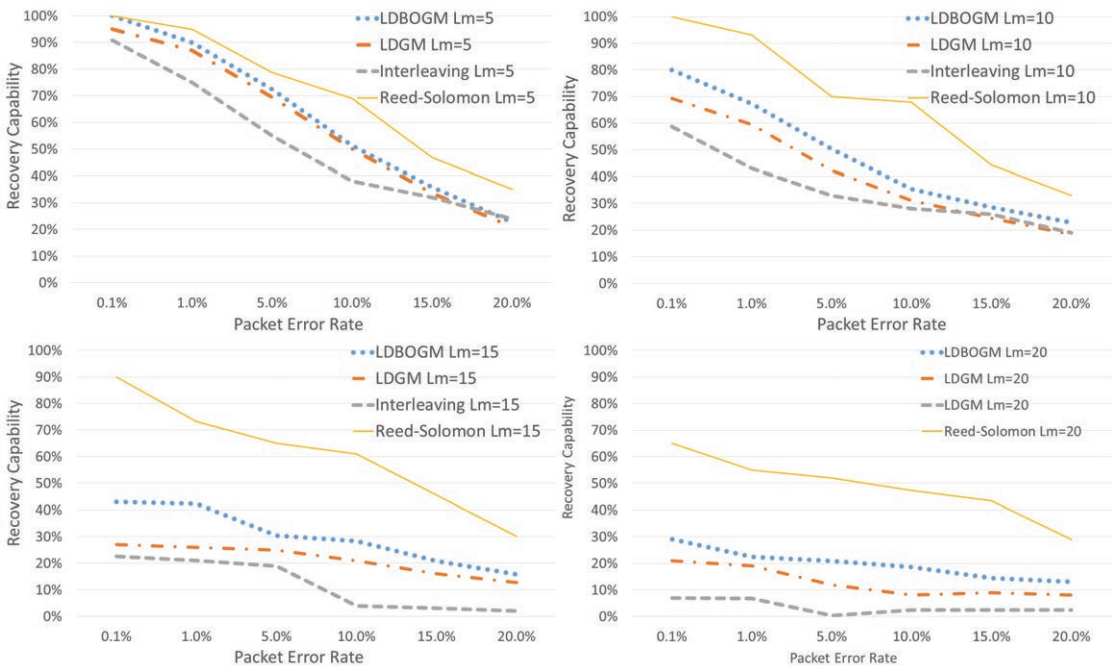


Figure 5.4.2: Comparison results. Percentage of recovered packets for L_m =5, 10, 15, and 20 and PER = 0.1, 1, 5, 10, 15, and 20 %

The results are presented in Figure 5.4.2. They show that the proposed algorithm algorithm outperforms the classical LDGM code in all the cases. In addition, they are more robust than the interleaving XOR-based scheme. Finally, one can see that for average burst lengths $L_m > 10$, the performance of the LDBOGM, LDGM, and interleaving decrease significantly compared to one of the Reed-Solomon scheme. This behavior is motivated by two factors: (i) the design of those algorithms is not optimal for burst lengths greater than $L_m = 10$ (for LDBOGM matrices the refinement process has been tuned for a window of $L_m = 10$); (ii) the Reed-Solomon scheme selected as reference uses a larger value of $n - k$. Regarding the size of the analysis window of the algorithm ($L_m = 10$), it was determined empirically. Several tests with different window sizes have been conducted. Short windows did not provide good results for the modified matrices since little columns were involved in the analysis, but long windows did not provide good results either. This might be due to the fact that considering a wide neighborhood at the end of the process changes due to different columns tend to void. Nevertheless, LDBOGM again outperforms all the approaches, with the only exception of the Reed-Solomon codes.

5.4.2.3 Complexity analysis: LDBOGM code vs. Reed-Solomon codes

In this Section, it is shown that, although Reed-Solomon codes perform better, LDBOGM code has better computing performance.

In [57], the authors show that Reed-Solomon codes may have a good performance in terms of complexity requirements, when carefully designed. In this sense, the authors refer to RFC 5510 [30], where the encoding complexity, in the case of the pre-computation of the generator matrix, involves k operations per repair element (vector-matrix multiplication). For the LDGM codes, considered in this Chapter, the number of operations equals $w_r - 1$, which is far lower than k.

Moreover, regarding the decoding process, in [57], the authors state that, although the decoding speeds are reasonable or high for Reed-Solomon codes, when it comes to small blocks, they are still lower than those of an LDPC-staircase code, which in turn are more complex than the ones used in this work.

In order to estimate the gain in performance that can be achieved using LDBOGM codes with respect to Reed-Solomon codes, the encoding and the decoding time for both schemes, for given values of n, k, and for a fixed packet size have been computed. For the Reed-Solomon implementation, the free codec provided by [84] implemented in C++ has been used. Moreover, the Reed-Solomon codec is set to its shortened version, selecting $n = 100$ and $k = 80$, since these values have been used in the experiments with the LDBOGM codes. The packet size has been set to 1024 bytes. The time measurements have been carried out in a PC with an Intel Core i7-3540 @ 3 GHz. The encoding time for 1024 bytes packets, $n = 100$ and $k = 80$, is shown in Table 5.4.3.

Table 5.4.3: Encoding time for Reed-Solomon codes and LDBOGM codes.

LDBOGM	Reed-Solomon	Performance gain
0.009 ms	0.048 ms	81 %

In order to estimate the decoding time of the algorithms, the transmission of 300 groups of $n = 100$ packets using four different channel models (average burst lengths $L_m = 5$, 10, 15, and 20, and a PER = 1%) has been simulated. These channel models are a subset of the channel models used in the general experiments. In these simulations, each time a loss has occurred, the decoding time for each protection scheme has been measured.

The presented results are the average value per group of $n = 100$ packets, as depicted in Table 5.4.4.

Table 5.4.4: Decoding time for Reed-Solomon codes and LDBOGM codes, PER=1 %

	$L_m = 5$	$L_m = 10$	$L_m = 15$	$L_m = 20$
LDBOGM	4.535 ms	6.152 ms	5.891 ms	7.862 ms
Reed-Solomon	8.950 ms	9.910 ms	12.734 ms	10.623 ms
Performance gain	49 %	38 %	54 %	26 %

As can be observed, in both cases (encoding and decoding) gains in terms of computing time have been obtained using the LDBOGM code: 81% for the encoding time, from 26% up to 54% for the decoding time.

5.5 Conclusions

In this Chapter, a novel algorithm algorithm has been proposed: the burst-oriented modifying algorithm. It allows boosting the recovery capability of LDPC codes against bursty packet losses, in contrast to the original randomly generated H matrix, which is oriented to memoryless channels.

Moreover, in this Chapter, small values of k (small information blocks) are used, in order to fulfil low transmission latency requirements for time-sensitive communications.

The proposed approach is based on a novel analysis that evaluates the recovery capability of each part of matrix H, with the aim of finding the weakest and strongest parts against bursts of lost packets. This analysis has been made by defining a local parameter, the CRM, column-based recovery measurement, and a global parameter, the $GRM(H)$, global recovery measurement.

Once the different parts of the matrix H are identified (minima and maxima of CRM), the second step of the proposed algorithm algorithm is to modify the weakest parts of the matrix H in order to make these parts stronger against bursts of lost packets, with the aim of improving the $GRM(H)$ value.

Finally, in order to assess the effectiveness of the algorithm, two types of experimental tests have been carried out. The first one consists in the generation of 50 different LDGM matrices, which have been modified using the algorithm to generate the corresponding LDBOGM matrices. Afterwards, the recovery capability for each pair of matrices (original and modified), for four different bursty channels in an RTP transmission, has been evaluated. It has been demonstrated that LDBOGMs obtain better results in almost all the experiments. In the second test, it has been selected the modified matrix that has obtained the best average recovery capabilities in the analysis presented in the previous Section (LBOGM) and compared it with (i) the original matrix, (ii) an XOR-based interleaved code, and (iii) a Reed-Solomon scheme. After a simulated RTP transmission

in 24 different bursty channels, it can be stated that the algorithm outperforms the classical LDGM codes and the interleaving XOR-based scheme. In addition, the recovery capabilities of the LDBOGM code approaches the values reached by the Reed-Solomon codes in some cases.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, the topic of the protection of a multimedia packetized stream from losses has been attempted. Hence, in order to find solutions to this problem, two novel protection architectures has been presented. These architectures have evaluated different FEC-based schemes, since, due to the time-sensitive nature of the considered communication, the retransmission of the lost packets is not possible (i.e. ARQ-based protocols). A theoretical study of FEC codes has been carried out and the characteristics and the properties of systematic block codes, particularly suited for this scenario, have been deeply analyzed. Several families of this type codes has been investigated, and their main advantages and main debilities have been underlined and discussed, with the aim of finding their most appropriate configuration.

Another important step, that has been considered in this work, is the characterization of the transmission scenario. Two main point of views have been contemplated: the first one is the protocols combination used in real time multimedia communications, whereas, the second one is represented by the statistical proprieties of the channel.

The combination of protocols, considered in this thesis, is based on the encapsulation of IP, UDP, and RTP. This type of communication architecture is centered in the use of FEC schemes at the Application Layer (RTP protcols). A deep discussion about the application of classical block codes to the Application Layer for packetized stream has been presented. Moreover, a relevant part of this work has been the design of the structure of the protection architecture taking into account the restrictions given by the RFCs.

The transmission system has not only been studied from the protocols point of view, but also from the unreliable nature of the communication channel: it is a Packet Erasure Channel where erasures are generated as bursts of lost packets. The evaluation of the statistical characteristics of the channel is a necessary step to design the protection architecture presented in this thesis.

The previous analysis has pointed out several limitations of classical FEC codes in relation with the IP-channel. Hence, the core of the thesis has been focused on optimizing

existing no patented codes, on the one hand from a computational point of view, since the considered communication is time sensitive, and, on the other hand, taking into account the statistical characteristics of the bursty channels.

For that reason, in this thesis, two different FEC architectures for the protection of multimedia packets at the Application Layer have been proposed. The first one permits to handle Reed-Solomon codes within a protection scheme intended for RTP transmission of multimedia contents: the inter-packet symbol approach. This approach is based on a novel bits organization of the symbols, that permits to exploit better the recovery capability of Reed-Solomon codes against bursty packet losses. The performance of inter-packet approach has been analyzed in terms of computational complexity versus recovery capability. Additionally, the comparison with other schemes, proposed in the literature and that follow an intra-packet symbol approach, has been done. The theoretical analysis has shown that our approach allows the use of a smaller size of the Galois Fields size compared to other solutions. This lower size results in a decrease of the required computational cost while keeping a comparable recovery capability, as shown in the experimental transmission tests.

The second proposed strategy is based on LDPC codes. Thanks to a novel proposed algorithm, the burst-oriented modifying algorithm, this type of codes are optimized to be used in memory channels, in contrast to the original randomly generated H matrix, which is oriented to memoryless channels. Moreover, small information blocks have been used, in order to fulfill low transmission latency requirements for time-sensitive communications.

This approach is based on a novel analysis that evaluates the recovery capability of each part of matrix H, with the aim of finding the weakest and strongest parts against bursts of lost packets. Once the different parts of the matrix H are identified, the second step of the proposed algorithm is to modify the weakest parts of the matrix H in order to make these parts stronger against bursts of lost packets, generating Low-Density Burst-Oriented Generator-Matrices (LDBOGM) .

In order to assess the effectiveness of the proposed algorithm, experimental tests have been carried out. After a simulated RTP transmission in 24 different bursty channels, we state that our algorithm outperforms the classical LDGM codes and the interleaving XOR-based scheme. In addition, the recovery capabilities of the LDBOGM code approaches the values reached by the Reed-Solomon codes in some cases.

The work in this thesis has resulted in several publications [1] [2] [3] [4] [5] [6].

6.2 Future work

Since the improvements of the architectures introduced in this thesis only take into account the recovery capability as the number of lost packets to recovered packets ratio, it would be interesting to evaluate the results calculated from a different point of view. In particular, as the transmitted data is basically time-sensitive video contents, such as video conference, it will be very useful to evaluate some metric, in order to quantify, on the one hand, the enhancement of the quality of the video, before and after the use of the

recovering architectures, and on the other hand, comparing it to the classical versions of the employed FEC codes. The metrics can be either objective ones, over the quality of the video, or subjective ones.

Moreover, smart protections mechanism should be investigated and applied to the architectures presented in this thesis. Also for this type of improvements, the results should be calculated as the percentage of recovered packets against lost packets and through other metrics related to the objective and subjective quality of the decoded video contents.

Finally, a more challenging future work is represented by the investigation of FEC codes families that are different from block codes.

The general ideas proposed in this Section, that indicate some possible future directions of this work, can be summarized in the following points:

- The design a transmission system where the Sequence Number of lost packet and recovered packets are saved. The two compressed video streams have to be decoded with the lost information at the receiver and the recovery capability has to be measured with objective metrics. This can be done, for example, by comparing the Peak Signal-to-Noise Ratio (PSNR) of the two decoded videos.

- The use of other types of metrics in order to evaluate the usefulness of the schemes proposed in this thesis. Hence, by using the method of the previous point, it would be interesting to dispose several users that can give a value of the Quality of the Experience (QoE) of the two decoded videos: the video with losses and the video after the applying of the recovery operations.

- The application of smart protection mechanisms, as Unequal Error Protection (UEP), to the packet stream, with the aim of increasing the recovery probability of those packets that transport important part of the information (i.e. I-frames of a encoded video).

- The design of dynamic FEC architectures that changes the code parameters depending on the condition of the channel.

- The optimization of other types of LDCP codes, the triangle and staircase versions (see Section 2.4.5.1 and RFC 5170 [50]). In particular the second one are very interesting, as recent works show (i.e. [85]).

- The investigation of other families of codes, as rateless codes, in particular Raptors/RaptorsQ codes [40] [41], and the study of the application of Convolutional Codes based on sliding encoding windows, that have not yet been considered for AL-FEC schemes, except for this technical report [86].

Appendix A

An Initial Evaluation of Video Response over DSL with LDGM Codes

A.1 Introduction

This Appendix considers an application-layer (AL) FEC solution to the problem of video packet loss on Digital Subscriber Lines (DSL), which is the dominant broadband access network for residential users with 364.1 million links in 2012 [87]. If packets are lost, owing to video-coding dependencies video streams may be disrupted for up to 500 ms, the duration of a typical Group of Pictures (GoP) [88].

This chapter considers the relevance of LDGM codes for video communication with shorter block lengths.

A.2 PSNR evaluation framework

In order to provide a realistic evaluation of video distortion, tests used the reference video sequence *Football*, with plenty of motion activity, which increases the temporal compression coding dependencies. In order to judge the video distortion, a video trace was fed into a numerical simulator (refer to Figure A.2.1) where ADSL packetization took place. After numerical simulation, data from the ADSL packets judged lost were removed from the compressed video bitstream, prior to passing through the H.264/Advanced Video Coding (AVC) [89] decoder. The resulting bitstreams (before and after LDGM repair) were compared to the YUV video input to determine the Peak Signal-to-Noise Ratio (PSNR).

To allow the gain from combining FEC with built-in error resiliency, High Efficiency Video Coding (HEVC) was not used as it has limited support for error resiliency. Instead the video sequence was encoded with the H.264/AVC JM 14.2 codec in Common

Intermediate Format (352 × 288 pixels/frame) at 30 frames/s with a Constant Bit Rate (CBR) of 1 Mbps. The frame structure was an initial intra-coded frame followed by all predictively-coded P-frames. 2% intra-coded macroblocks (MBs) were included in the P-frames to guard against temporal error propagation. The IPPP. . . frame structure with intra-coded MBs included, is also suitable for streaming to mobile devices, as there is reduced computation because bi-predictive B-frames are no longer employed. Channel switching, for which periodic I-frames are useful, is not expected in a telemedicine or video-conferencing application. Data partitioning was also turned on at the codec as an additional form of error resilience, with constrained intra prediction also configured. These video settings conform to the recommendations of [90].

A.3 Impact upon video distortion from LDGM

In the experiments, the code was a regular LDGM code with degree three. The parity-check matrix was created by using the classic stochastic algorithm [80]. Decoding was based on the belief propagation iterative algorithm.

In experiments in this Section, ADSL was assumed with small packet sizes of 50 B and 100 B for each of two sets of tests respectively. However, the results from testing with a larger 100 B packet size are also included in this Section, bringing the packet size closer to that of an MPEG2-TS packet. For 100 B packets a downstream bitrate of 10 Mbps was configured, with a per-packet link latency of around 100 ms. For 50 B packets and two-way communication a 1 Mbps effective datarate was assumed, with a per-packet link latency of around 10 ms. As previously, the PER was set 1% [91], though with burst lengths of 8 and 10 packets for the 50 B and 100 B packets respectively.

To counter error bursts, the packet block size was set to $k = 300$, $k = 400$. $n - k$, the number of redundant packets, was somewhat reduced to 9% of the whole. The latency budget remains well below the previously mentioned 1000 packets length of large block coding schemes. Table A.3.1 shows five sample runs each with a different seed and the resulting mean PSNR. (The code seed was set to 40, 50, 70, 80, and 90 for 1–5 respectively.) The mean gain after application of LDGM was between 1 and 2 dB with the video PSNR approaching a level suitable for broadcast. Interestingly from the point of view of latency, increasing the block size does not necessarily lead to a reduction in video distortion.

Before FEC	**1**	**2**	**3**	**4**	**5**	**Mean PSNR (dB)**
$k = 300$	35.53	35.08	38.63	36.14	37.37	36.55
$k = 400$	34.46	37.66	38.54	36.40	39.00	37.21
After FEC	**1**	**2**	**3**	**4**	**5**	
$k = 300$	37.33	39.00	39.00	37.57	37.84	38.15
$k = 400$	36.91	37.76	39.00	37.64	39.00	38.06

Table A.3.1: Objective video PSNR for 50 B packets before and after FEC.

For the larger packet size and the greater bandwidth of Table A.3.2, the video distortion reduction is more consistent and is 3–4 dB. The consistency is due to a constant code seed of 50 throughout. Notice that in view of the larger 100 B packet size the block sizes are decreased. Again a larger block size appears not to lead to an advantage. This effect may be linked to the pattern of packet burst erasures. Comparing the 100 B PSNR gain to that of 50 B packets, for the latter the FEC gain appears to have saturated, suggesting a reduced FEC rate is possible.

Before FEC	**1**	**2**	**3**	**4**	**5**	**Mean PSNR (dB)**
$k = 200$	34.53	33.95	34.91	34.93	35.64	34.79
$k = 300$	36.54	36.41	33.81	34.69	32.84	34.86
After FEC	**1**	**2**	**3**	**4**	**5**	
$k = 200$	38.22	36.67	39.00	39.00	38.86	38.35
$k = 300$	38.85	36.55	39.00	38.88	37.35	38.13

Table A.3.2: Objective video PSNR for 100 B packets before and after FEC.

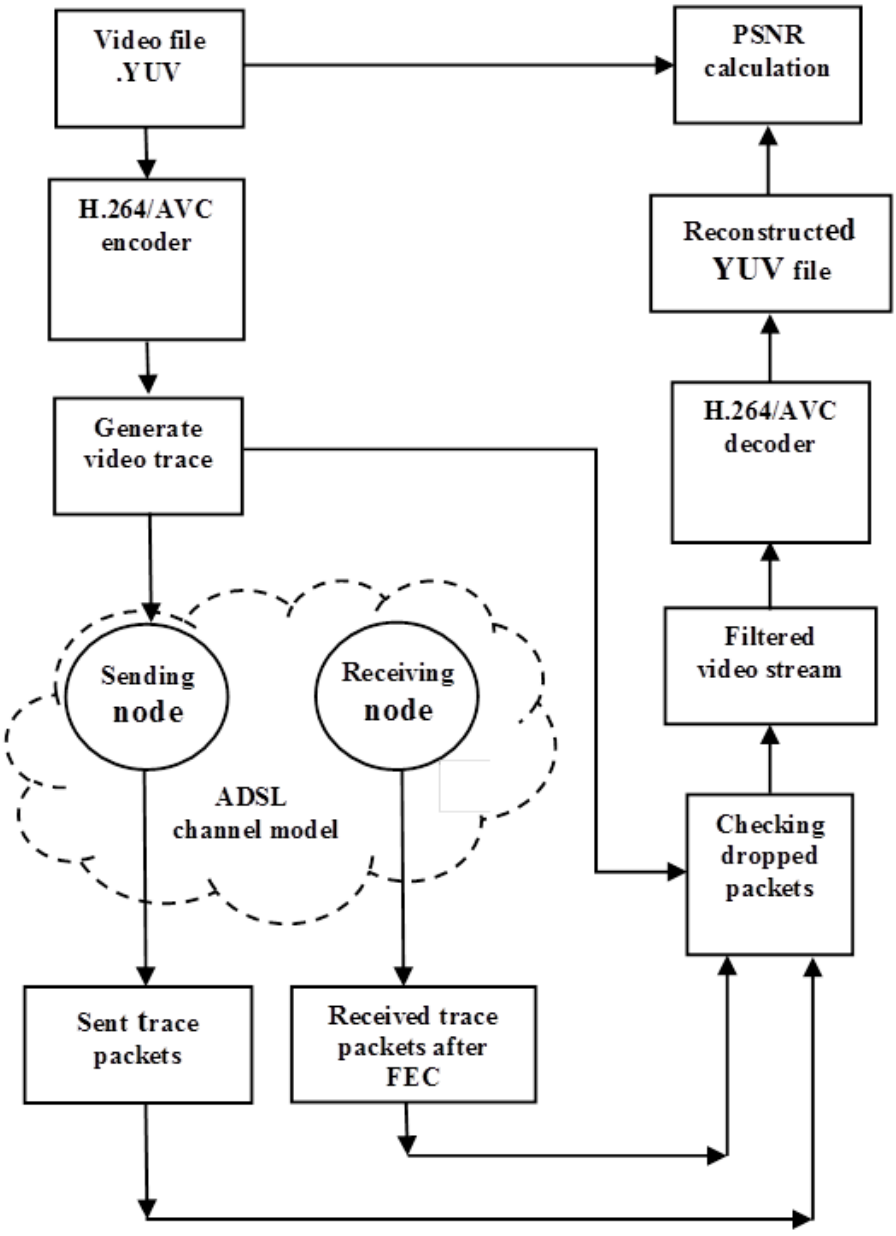


Figure A.2.1: PSNR evaluation framework.

Bibliography

[1] F. Casu, J. Cabrera, F. Jaureguizar, and N. García, "Inter-packet symbol approach to reed-solomon fec codes for rtp-multimedia stream protection," in *2011 IEEE Symposium on Computers and Communications (ISCC)*, June 2011, pp. 49–54.

[2] ——, "A protection scheme for multimedia packet streams in bursty packet loss networks based on small block low-density parity-check codes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, p. 187, 2015. [Online]. Available: http://dx.doi.org/10.1186/s13638-015-0402-6

[3] ——, "Low latency ldgm code for multimedia-packet stream in bursty packet loss networks," in *2012 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2012, pp. 63–64.

[4] L. Al-Jobouri, F. Casu, M. Fleury, and J. Cabrera, "Effective al-fec with ldgm for interactive video over dsl," in *2015 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2015, pp. 358–359.

[5] F. Casu, J. Cabrera, L. Al-Jobouri, and M. Fleury, "Video over dsl with ldgm codes for interactive applications," in *2015 7th Computer Science and Electronic Engineering Conference (CEEC)*, Sept 2015, pp. 161–166.

[6] L. Al-Jobouri, F. Casu, M. Fleury, and J. Cabrera, "Video over dsl with ldgm codes for interactive applications," *Computers*, vol. 5, no. 2, p. 9, 2016.

[7] G. Fairhurst and L. Wood, "Advice to link designers on link automatic repeat request (arq)," Internet Requests for Comments, RFC Editor, BCP 62, August 2002, techn. report. RFC 3366 (2002). [Online]. Available: http://www.rfc-editor.org/rfc/rfc3366.txt

[8] S. B. Wicker, *Error control systems for digital communication and storage*. Prentice hall Englewood Cliffs, 1995, vol. 1.

[9] V. Miguel, J. Cabrera, F. Jaureguizar, and N. García, "High-definition video distribution in 802.11g home wireless networks," in *IEEE International Conference on Consumer Electronics (ICCE), 2011*, January 2011, pp. 211 –212.

[10] S. Benedetto and E. Biglieri, *Principles of digital transmission: with wireless applications*. Springer Science & Business Media, 1999.

[11] P. Sweeney, *Error control coding: from theory to practice*. John Wiley & Sons, Inc., 2002.

[12] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[13] R. Garello, "Lecture notes in tecniche di protezione dell'informazione," 2009, dipartimento di Elettronica, Politecnico di Torino (2009). [Online]. Available: http://www.tlc.polito.it/garello/protezione/protezione.html

[14] M.-F. Tsai, T.-C. Huang, C.-K. Shieh, and K.-C. Chu, "Dynamical combination of byte level and sub-packet level fec in harq mechanism to reduce error recovery overhead on video streaming over wireless networks," *Comput. Netw.*, vol. 54, no. 17, pp. 3049–3067, Dec. 2010.

[15] S. Lin and D. J. Costello, *Error Control Coding, Second Edition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.

[16] R. H. Morelos-Zaragoza, *The Art Of Error Correcting, 2nd Edition*. Chichester, UK: John Wiley & Sons, 2006.

[17] A. J. Viterbi and J. K. Omura, *Principles of digital communication and coding*. Courier Corporation, 2013.

[18] R. H. Morelos-Zaragoza, *The art of error correcting coding*. John Wiley & Sons, 2006.

[19] A. Li, "Rtp payload format for generic forward error correction," Internet Requests for Comments, RFC Editor, RFC 5109, December 2007, techn. report. RFC 5109 (2007). [Online]. Available: http://www.rfc-editor.org/rfc/rfc35109.txt

[20] M. Watson, A. Begen, and V. Roca, "Forward error correction (fec) framework," Internet Requests for Comments, RFC Editor, RFC 6363, October 2011, techn. report. RFC 6363 (2011). [Online]. Available: http://www.rfc-editor.org/rfc/rfc6363.txt

[21] B. Sklar, *Digital Communications: Fundamentals and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[22] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society of Industrial and Applied Mathematics*, vol. 8, no. 2, p. 300–304, 06/1960 1960. [Online]. Available: http://www.jstor.org/pss/2098968

[23] S. B. Wicker, *Reed-Solomon Codes and Their Applications*. Piscataway, NJ, USA: IEEE Press, 1994.

[24] D. Chase, "Class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 170–182, Jan 1972.

[25] "Transmission of professional mpeg-2 transport streams over ip networks," Pro-Meg Forum, Code of Practice #3 release 2, July 2004, pro-Meg Forum, Techn. report., Code of Practice #3 release 2 (2004).

[26] C. Díaz, J. Cabrera, F. Jaureguizar, and N. García, "An extension to the pro-mpeg cop3 codes for unequal error protection of real-time video transmission," in *2015 IEEE International Conference on Image Processing (ICIP)*, Sept 2015, pp. 927–931.

[27] S. Wicker and V. Bhargava, *Reed-Solomon codes and their applications.* Wiley-IEEE Press, 1999.

[28] B. Sklar, "Introduction to reed-solomon codes," *Digital Communications: Fundamentals and applications*, 2001.

[29] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, 1997.

[30] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes," RFC 5510 (Proposed Standard), Tech. Rep. 5510, April 2009, techn. report. RFC 5510 (2009). [Online]. Available: http://www.rfc-editor.org/rfc/rfc5510.txt

[31] M. Sudan, "Decoding of reed solomon codes beyond the error-correction bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180 – 193, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0885064X97904398

[32] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.

[33] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *Information Theory, IEEE Transactions on*, vol. 45, no. 2, pp. 399–431, Mar 1999.

[34] A. Shokrollahi, *LDPC Codes: An Introduction.* Basel: Birkhäuser Basel, 2004, pp. 85–110. [Online]. Available: http://dx.doi.org/10.1007/978-3-0348-7865-4_5

[35] W. Ryan, *An introduction to LDPC codes.* CRC Press, 2005, ch. 10.

[36] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 638–656, Feb 2001.

[37] A. Nafaa, T. Taleb, and L. Murphy, "Forward error correction strategies for media streaming over wireless networks," *Communications Magazine, IEEE*, vol. 46, no. 1, pp. 72–79, 2008.

[38] D. J. MacKay, "Fountain codes," *IEE Proceedings-Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[39] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[40] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "Raptor forward error correction scheme for object delivery," Internet Requests for Comments, RFC Editor, RFC 5053, October 2007, techn. report. RFC 5053 (2007). [Online]. Available: http://www.rfc-editor.org/rfc/rfc5053.txt

[41] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "Raptorq forward error correction scheme for object delivery," Internet Requests for Comments, RFC Editor, RFC 6330, August 2011, techn. report. RFC 6330 (2011). [Online]. Available: http://www.rfc-editor.org/rfc/rfc6330.txt

[42] P. Maymounkov, "Online codes," Technical report, New York University, Tech. Rep., 2002, techn. report. New York University (2002).

[43] P. Maymounkov and D. Mazières, *Rateless Codes and Big Downloads*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 247–255. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45172-3_23

[44] F. Didier, "Efficient erasure decoding of reed-solomon codes," *CoRR*, vol. abs/0901.1886, 2009. [Online]. Available: http://arxiv.org/abs/0901.1886

[45] V. V. Zyablov and M. S. Pinsker, "Decoding complexity of low-density codes for transmission in a channel with erasures," *Problemy Peredachi Informatsii*, vol. 10, no. 1, pp. 15–28, 1974.

[46] G. A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, pp. 71–78, 1982. [Online]. Available: http://dx.doi.org/10.1007/BF02579283

[47] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep 1981.

[48] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–, Aug 1996.

[49] N. Wiberg, H. A. Loeliger, and R. Kotter, "Codes and iterative decoding on general graphs," in *Proceedings of 1995 IEEE International Symposium on Information Theory*, Sep 1995, pp. 468–.

[50] V. Roca, C. Neumann, and D. Furodet, "Low density parity check (LDPC) staircase and triangle forward error correction (FEC) schemes," RFC 5170, Tech. Rep., 2008, techn. report. RFC 5170 (2008). [Online]. Available: http://www.rfc-editor.org/rfc/rfc5170.txt

[51] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[52] V. Roca, Z. Khallouf, and J. Laboure, "Design and evaluation of a low density generator matrix (ldgm) large block fec codec," in *in Fifth International Workshop on Networked Group Communication (NGC 2003)*, 2003.

[53] C. Davey, Matthew, "Error-correction using low-density parity-check codes," Ph.D. dissertation, University of Cambridge, 1999, phD thesis, University of Cambridge (1999).

[54] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, Nov 1996.

[55] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '97. New York, NY, USA: ACM, 1997, pp. 150–159. [Online]. Available: http://doi.acm.org/10.1145/258533.258573

[56] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1723–1731, Nov 1996.

[57] V. Roca, M. Cunche, C. Thienot, J. Detchart, and J. Lacan, "Rs ldpc-staircase codes for the erasure channel: Standards, usage and performance," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, Oct 2013, pp. 638–644.

[58] V. Roca and C. Neumann, "Design, Evaluation and Comparison of Four Large Block FEC Codecs, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon Small Block FEC Codec," INRIA, Rapport de recherche RR-5225, 2004. [Online]. Available: http://hal.inria.fr/inria-00070770

[59] M. Cunche and V. Roca, "Improving the decoding of ldpc codes for the packet erasure channel with a hybrid zyablov iterative decoding/gaussian elimination scheme," Ph.D. dissertation, INRIA, 2008.

[60] K. R. Rao, Z. S. Bojkovic, and D. A. Milovanovic, *Multimedia Communication Systems: Techniques, Standards, and Networks.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[61] M. v. d. Schaar and P. A. Chou, *Multimedia over IP and Wireless Networks: Compression, Networking, and Systems.* Orlando, FL, USA: Academic Press, Inc., 2007.

[62] A. Talari, S. Kumar, N. Rahnavard, S. Paluri, and J. Matyjas, "Optimized cross-layer forward error correction coding for h.264 avc video transmission over wireless channels," *EURASIP Journal on Wireless Communications and*

Networking, vol. 2013, no. 1, 2013. [Online]. Available: http://dx.doi.org/10.1186/1687-1499-2013-206

[63] C. Lamoriniere, A. Nafaa, and L. Murphy, "Dynamic switching between adaptive fec protocols for reliable multi-source streaming," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, 2009, pp. 1–6.

[64] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 3550, Tech. Rep., 2003, techn. report. RFC 3550 (2003). [Online]. Available: http://www.rfc-editor.org/rfc/rfc3550.txt

[65] C. Díaz, J. Cabrera, F. Jaureguizar, and N. García, "A video-aware fec-based unequal loss protection system for video streaming over rtp," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 523–531, 2011.

[66] T. Paila, R. Walsh, M. Luby, V. Roca, and R. Lehtonen, "Flute - file delivery over unidirectional transport," Internet Requests for Comments, RFC Editor, RFC 6726, November 2012, techn. report. RFC 6726 (2012).

[67] "Network model for evaluating multimedia transmission performance over internet protocol," ITU-T Recommendation G.1050, Tech. Rep., November 2007, techn. report., ITU-T Recommendation G.1050 (2007).

[68] M. Johanson, "Adaptive forward error correction for real-time internet video," in *Proc. PV 2003, 13th Packet Video Workshop, Nantes, France*. Citeseer, April 2003.

[69] K. Matsuzono, J. Detchart, M. Cunche, V. Roca, and H. Asaeda, "Performance analysis of a high-performance real-time application with several al-fec schemes," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, 2010, pp. 1 –7.

[70] S. Galanos, O. Peck, and V. Roca, "Rtp payload format for reed solomon fec," Internet-Draft, March 14, 2011, Expires: September 15, 2011, Tech. Rep.

[71] A. Bennatan and D. Burshtein, "On the application of ldpc codes to arbitrary discrete-memoryless channels," *Information Theory, IEEE Transactions on*, vol. 50, no. 3, pp. 417–438, March 2004.

[72] S. Johnson, "Burst erasure correcting ldpc codes," *Communications, IEEE Transactions on*, vol. 57, no. 3, pp. 641–652, 2009.

[73] J. Xu, L. Chen, L. Zeng, L. Lan, and S. Lin, "Construction of low-density parity-check codes by superposition," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 243–251, 2005.

[74] Y. Y. Tai, L. Lan, L. Zeng, S. Lin, and K. Abdel-Ghaffar, "Algebraic construction of quasi-cyclic ldpc codes for the awgn and erasure channels," *Communications, IEEE Transactions on*, vol. 54, no. 10, pp. 1765–1774, Oct 2006.

[75] G. Hosoya, H. Yagi, T. Matsushima, and S. Hirasawa, "A modification method for constructing low-density parity-check codes for burst erasures," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E89-A, no. 10, pp. 2501–2509, Oct. 2006.

[76] E. Paolini and M. Chiani, "Construction of near-optimum burst erasure correcting low-density parity-check codes," *Communications, IEEE Transactions on*, vol. 57, no. 5, pp. 1320–1328, May 2009.

[77] M. Fossorier, "Universal burst error correction," in *2006 IEEE International Symposium on Information Theory*, July 2006, pp. 1969–1973.

[78] ——, "Hybrid burst erasure correction of ldpc codes," *Communications Letters, IEEE*, vol. 13, no. 4, pp. 260–261, April 2009.

[79] E. Martinian and C.-E. Sundberg, "Burst erasure correction codes with low decoding delay," *Information Theory, IEEE Transactions on*, vol. 50, no. 10, pp. 2494–2502, Oct 2004.

[80] V. Roca, C. Neumann, M. Cunche, and J. Laboure, "LDPC-Staircase/LDPC-Triangle/LDGM Codec Reference Implementation," 2005. [Online]. Available: http://planete-bcast.inrialpes.fr

[81] "OpenFEC Project." [Online]. Available: http://openfec.org

[82] C. Díaz, J. Cabrera, F. Jaureguizar, and N. García, "Adaptive protection scheme for mvc-encoded stereoscopic video streaming in ip-based networks," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*, 2012, pp. 1–6.

[83] A. Begen, "Error control for iptv over xdsl networks," in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, 2008, pp. 632–637.

[84] "Schifra Reed-Solomon error correcting code library." [Online]. Available: www.schifra.com

[85] F. Mattoussi, "Design and optimization of al-fec codes: the gldpc-staircase codes," Ph.D. dissertation, Universite De Grenoble, 2014, phD thesis, Universite De Grenoble (2014).

[86] V. Roca, B. Teibi, C. Burdinat, T. Tran, and C. Thienot, "Block or Convolutional AL-FEC Codes? A Performance Comparison for Robust Low-Latency Communications," Nov. 2016, working paper or preprint. [Online]. Available: https://hal.inria.fr/hal-01395937

[87] F. Vanier, "World broadband statistics: Q1 2009," Point Topic Ltd., June, Tech. Rep., 2009.

[88] M. Ghanbari, *Standard codecs: Image compression to advanced video coding*. Iet, 2003, no. 49.

[89] S. Wenger, "H.264/avc over ip," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, July 2003.

[90] L. Al-Jobouri, M. Fleury, and M. Ghanbari, "Engineering wireless broadband access to {IPTV}," *Journal of Visual Communication and Image Representation*, vol. 25, no. 7, pp. 1493 – 1506, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1047320314001126

[91] M. Luby, T. Stockhammer, and M. Watson, "Iptv systems, standards and architectures: Part ii - application layer fec in iptv services," *Comm. Mag.*, vol. 46, no. 5, pp. 94–101, May 2008.

Nomenclatura

k	the length, number of symbols, of an information (or data) vector
n	the length, number of symbols, of a code vector;
m	the number of bits of a symbol
R_c	rate code, k to n ratio
H_k	Hamming space of dimension k
$\mathbf{v}$	information, or data, vector
$\mathbf{c}$	code vector
G	generator matrix
w_k	Hamming weight
d_H	Hamming distance
d_m	minimum Hamming distance
H	parity matrix
w_c	number of 1s per column in a matrix
w_r	number of 1s per row in a matrix
L_m	average burst length
PEC	packet error rate in percentage
t_e	encoding time
t_d	decoding time

N_o^{intra}	the number of times the encoding operations for intra scheme
N_o^{inter}	the number of times the encoding operations for inter scheme
t_e^{tot}	total encoding time for a squence
C_e	constant that reflects the speed of the encoding system
C_d	constant that reflects the speed of the decoding system
M	the total number of RTP-media packets needed to stream a media content
L	the number of bits of each information string
$Nor\ t_e^{tot}$	total encoding time in normalized units
CRM(j)	The recovery capability of column j (column-based recovery measurement) as the number of recovered bursts
B_j^l	a burst of length l, $l \in (2, n-k)$, starting in packet j, $j \in (0, k-1)$
GRM(H)	the Global Recovery Measurement as $\sum_{j=0}^{k-1}$ CRM(j)